UNIVERSITA' DEGLI STUDI DI FERRARA

DOTTORATO DI RICERCA IN SCIENZE DELL'INGEGNERIA

Ciclo XXIII

COORDINATORE Prof. Stefano Trillo

NUOVI PROTOCOLLI PER GESTIONE DATI E COMUNICAZIONE REAL-TIME IN AMBITO AGRICOLO

Settore Scientifico Disciplinare ING-INF/05

DOTTORANDO

Ing. Alfredo Revenaz

TUTORE INTERNO

Prof. Velio Tralli

TUTORE ESTERNO

Prof. Massimiliano Ruggeri

Anni 2008-2010

Alla mia famiglia, in particolare alla parte che deve ancora venire.

Capitolo 1

1.1 Premessa 17

1.2 Introduzione 17

1.3 Struttura dell'elaborato 18

Capitolo 2

2.1 Introduzione - ISO 19

2.2 ISO - cenni alla struttura 19

2.2.1 ISO - Assemblea Generale 20

2.2.2 ISO - Il Consiglio 20

2.2.3 ISO - Segreteria Centrale 20

2.2.4 ISO - Le Commissioni per lo Sviluppo delle Politiche 20

2.2.5 ISO - Le Commissioni permanenti del Consiglio 21

2.2.6 ISO - Gruppi di consulenza ad hoc 21

2.2.7 ISO - Consiglio Direttivo Tecnico (TMB) 21

2.2.8 ISO - Gruppi di consulenza strategica e tecnica e REMCO 22

2.2.9 ISO - Commissioni, Sottocommissioni Tecniche e Gruppi di Lavoro 23

2.2.10 ISO - TC23 - SC19 - WG1, WG3, WG5 23

2.2.11 Le attività di ricerca in accordo alle richieste del WG5 24

2.2.12 La nuova normativa 25

2.3 Approccio progettuale 26

2.4 Divisione del lavoro - i tre Task 27

2.4.1 Task 0 27

2.4.2 Task 1 28

2.4.3 Task 2 28

Capitolo 3

3.1 Premessa 29
3.2 Introduzione 29
3.3 Normazione e organismi transnazionali 30
3.4 Che cosa è ISOBUS 31
3.5 ISOBUS: La struttura della rete 32
3.6 Il Virtual Terminal 33
3.7 Gli Auxiliary Input 34
3.8 La TECU 37
3.9 Il Task Controller e il File Server 38
3.10 Il Sequence Controller 40
3.11 I sistemi di guida autonoma 42
3.12 La progettazione dei componenti e le aziende coinvolte 42
3.13 L'evoluzione dello standard per i prossimi anni 43
3.14 La previsione di mercato per l'immediato futuro 44

Capitolo 4

4.1 Task 0 47
4.2 Task 0 - Contesto attuale 47
4.3 Finalità progettuali 48
4.4 Schema generale Mobile Information System 50
4.4.1 Mobile Information Base 50
4.4.2 Terminali mobili 51
4.4.3 Stazioni mobili 51
4.4.4 Rete locale sensori 51
4.4.5 Interfacce grafiche remote 52
4.5 Schema Mobile Information Server 52
4.5.1 Mobile Information Server - caratteristiche e componenti 53

4.6 Proof of Concept - introduzione 53
4.7 Proof of Concept - premessa 54
4.7.1 Proof of Concept - Il software FLOSS 54
4.8 Proof of Concept - Definizione dell'ambiente di sviluppo e gestione 55
4.8.1 Il Sistema Operativo 56
4.8.2 Eclipse - Ambiente di Sviluppo Integrato 59
4.8.3 Il Relational Database Management System 60
4.8.3.1 Scelta del RDBMS 60
4.8.3.2 Caratteristiche di PostgreSQL 64
4.9 Proof of Concept - Il Database 66
4.9.1 Database - Gestione periferiche 67
4.9.2 Database - Gestione Dati 71
4.10 Proof of Concept - Software di gestione del Database 73
4.10.1 Il linguaggio di programmazione 73
4.10.2 Libreria aggiuntiva Postgres: libpq-fe.h 74
4.10.3 Applicativo realizzato 81
4.11 Proof of Concept - Terminale progettato e realizzato 84
4.11.1 Periferica GPS 85
4.11.2 Lo standard PC/104 86
4.11.3 La scheda in standard PC/104 87
4.11.4 Il software GPSD 89
4.11.5 Software applicativo del terminale 90
4.12 Task 0 - Conclusioni 94
4.13 Task 0 - Sviluppi futuri 95

Capitolo 5

5.1 Task 1 97

5.2 Task 1 - Alcune considerazioni e problemi 98
5.3 Cenni di crittografia moderna 99
5.3.1 Crittografia Simmetrica 100
5.3.2 Crittografia Asimmetrica 101
5.3.3 Algoritmi ibridi 102
5.3.4 Crittografia in ISOBUS - ulteriori problemi 103
5.4 ISOBUS - Comunicazione long range e gestione dati sicure - la soluzione. 104
5.4.1 I punti 1, 2 e 3 104
5.4.1.1 I protocolli di trasporto in ISOBUS 105
5.4.1.2 ISOBUS Transport Protocol 105
5.4.1.3 ISOBUS Extended Transport Protocol 109
5.4.2 Soluzione per i punti 1, 2 e 3 110
5.4.3 Soluzione per i punti 4 e 5 111
5.5 Task 1 - Conclusioni 112
5.6 Task 1 - Sviluppi futuri 113

Capitolo 6

6.1 Task 2 - Background 115
6.2 Task 2 - Introduzione 115
6.3 Task 2 - Visione e futuro a lungo termine 116
6.4 Task 2 - Il futuro a medio e breve termine 118
6.4.1 Cluster di veicoli collaborativi omogenei 119
6.4.2 Cluster di veicoli collaborativi eterogenei 121
6.5 Task 2 - Premesse sul contesto 123
6.6 Task 2 - Una scelta ‘Open’ 124
6.7 Task 2 - Protocolli di riferimento 125
6.7.1 TCP, UDP e IP 128

6.7.2 Transmission Control Protocol (TCP) 129
6.7.3 User Datagram Protocol (UDP) 130
6.7.4 Comparazione tra UDP e TCP 131
6.7.5 Accesso diretto al protocollo IP 132
6.8 Protocolli scelti 132
6.9 Latenza: incidenza del Sistema e del Software 133
6.9.1 Sorgenti di latenza 134
6.10 Latenza e concetto di priorità 138
6.10.1 Latenza e Sistemi Operativi: Linux Real-Time 138
6.10.2 Red Hat Enterprise MRG e derivati 141
6.10.3 Latenza e gestione del traffico di rete 143
6.10.4 Standard Linux kernel Traffic Management 144
6.10.5 Linux InterMediate Queuing device (IMQ) 146
6.10.6 Agro Kernel 149
6.11 Task 2 - Hardware 151
6.12 Il Protocollo proposto 153
6.12.1 Un singolo protocollo per differenti funzioni 153
6.12.2 Quadro generale 154
6.12.3 Alcune definizioni 155
6.12.4 Livelli di priorità 156
6.12.5 Definizioni su comunicazione ed errori 157
6.12.6 Metodi per confinamento di errori di comunicazione 158
6.12.7 Analisi di sicurezza 160
6.12.8 La struttura del pacchetto 163
6.12.9 Lo stesso protocollo per scopi differenti 164
6.12.10 Un set di protocolli - differenti funzionalità 166
6.13 Ambiente di sviluppo 167

6.13.1 Subversion 167
6.13.2 Octave 168
6.14 Strumenti software sviluppati: quadro generale 168
6.14.1 Software in C 170
6.14.1.1 Sender_2 170
6.14.1.2 Receiver_2 170
6.14.2 Octave scripts. 171
6.14.2.1 plotting_script.m 171
6.14.2.2 generate_test_sheet.m 172
6.14.2.3 header_def.txt 172
6.14.3 Script di Shell. 172
6.14.3.1 multiple_sender.sh 172
6.14.3.2 multiple_receiver.sh 172
6.14.3.4 pretestspider 172
6.14.3.5 testspider 173
6.14.3.6 remote_control.sh 173
6.15 Un ambiente di sviluppo e test distribuito 173
6.15.1 Imalab 174
6.15.2 Siti di test interni 176
6.15.3 Siti di test esterni 178
6.15.4 Piattaforme di test 178
6.16 Ambiente di test 179
6.16.1 Test effettuati 183
6.16.2 Test iniziali 183
6.16.3 Test iniziali con kernel RT. 186
6.16.4 Test recenti. 187
6.17 Discussione dei test. 191

6.18 Task 2: Conclusioni 191

6.19 Task 2: Sviluppi futuri 192

Capitolo 7

7.1 Pubblicazioni e prodotti 195

7.2 Conclusioni finali 195

Ringraziamenti

Riferimenti

Norme Internazionali 199

Articoli 199

Testi 201

In Internet 202

Appendice A

Schema Mobile Information System (MI System) 205

Schema Mobile Information Server (MI Server) 206

Appendice B

Struttura del Database 207

Appendice C

Kernel packet traveling diagram 208

Roadmap implementazione AEMCP protocol 209

Field test sheet 210

Appendice D

Glossario 211

Articoli

1. International Symposium on Industrial Electronics (ISIE) 2010 Bari. ISIE2010 proceedings p.3498-3504 ISBN/ISSN 978-1-4244-6391-6 213

2. International Symposium on Industrial Electronics (ISIE) 2011. Sottomesso il 31-12-2010. Accettato il 23-02-2011. Sarà presentato al congresso dal 27 al 30 giugno 2011. 220

Capitolo 1

Capitolo 2

Figura 2.1: Organigramma ISO 19

Capitolo 3

Figura 3.1: Struttura della rete ISOBUS e della rete Powertrain con le principali unità elettroniche di controllo 33

Figura 3.2: Virtual Terminal John Deere screenshot schermata di comando e monitoraggio di Sprayer 34

Figura 3.3: Consolle di comando FENDT con Auxiliary Input e relativa pagina di configurazione sul Virtual Terminal 35

Figura 3.4: Pagina di associazione tra Auxiliary Input disponibili e funzionalità comandabili da remoto in una trattrice FENDT 35

Figura 3.5: Pagina di richiamo delle associazione realizzate tra Auxiliary Input disponibili e funzionalità comandabili da remoto in una trattrice FENDT 36

Figura 3.6: Virtual terminal Kverneland con le due aree relative al controllo dell'attrezzo (alto) e al controllo di lavorazione mediante Task Controller (basso). 38

Figura 3.7: Applicazione su PC per la programmazione delle lavorazioni su task Controller ISOBUS e per visualizzazione dei dati da File Server 39

Figura 3.8: Applicazione su PC per la programmazione delle lavorazioni su task Controller ISOBUS e per visualizzazione dei dati da File Server 41

Figura 3.9: Marcia parallela di Harvester e trattrice con caricamento di materiale durante la marcia dei veicoli. 44

Capitolo 4

Figura 4.1: Albero delle più note distribuzioni GNU/Linux 56

Tabella 4.2: Limiti di alcuni noti RDBMS Open Source 60

Tabella 4.3: Caratteristiche degli RDBMS 61

Tabella 4.4: Principali funzionalità aggiuntive degli RDBMS 62

Figura 4.5: Database - Relazioni tabelle Tipi e Periferiche_inizializzate. 70

Figura 4.6: Database - Relazioni tabelle Periferiche_inizializzate e GPS 72

Figura 4.7: Garmin GPSMAP 76S 85

Figura 4.8: Scheda PC104 Icop Technology Inc. montata in contenitore 88

Figura 4.9: Terminale GPS: stampa dati a video 92

Figura 4.10: Terminale GPS: inserimento dati 93

Figura 4.11: Terminale GPS: inizializzazione nuovo terminale 94

Figura 4.12: Terminale GPS: inizializzazione nuovo tipo di terminale 94

Capitolo 5

Figura 5.1: Comunicazioni long range in ambito agricolo 97

Figura 5.2: Cifratura simmetrica 100

Figura 5.3: Cifratura asimmetrica 101

Tabella 5.4: ISOBUS - Motivi e codici di Connection Abort 108

Capitolo 6

Figura 6.1: Lo scenario Agricolo a medio termine 118

Figura 6.2: Lavorazione con gruppo di macchine omogenee 119

Figura 6.3: Lavorazione con gruppo di macchine eterogenee 122

Tabella 6.4: Lo stack ISO/OSI 128

Figura 6.5: Livelli di priorità del kernel real-time 140

Figura 6.6: Default output policy del Linux kernel 2.6 145

Figura 6.7: Default input policy del Linux kernel 2.6 147

Figura 6.8: Linux IMQ concept 148

Figura 6.9: Linux IMQ - Ingress ed Egress 148

Figura 6.10: Agro patch traffic priority 150

Figura 6.11: Diagramma funzionale di una porzione del kernel Linux 2.4 151

Tabella 6.12: Tabella Errori di comunicazione e contromisure 161

Tabella 6.13: Struttura del pacchetto incapsulato in UDP 162

Figura 6.14: Struttura del pacchetto incapsulato in IP e UDP 163
Figura 6.15: Schema generale della suite software sviluppata 169
Figura 6.16: Ambiente di sviluppo e test distribuito 173
Figura 6.17: Schema Imalab 174
Figura 6.18: Foto Imalab 176
Figura 6.19: Schema indoor test site 176
Figura 6.20: Foto indoor test site 177
Figura 6.21: Foto outdoor test site 178
Figura 6.22: Base run test - 1 stream - no FTP 184
Figura 6.23: Worst case test - 1 stream - with FTP 185
Figura 6.24: Receiver non RT Sender RT – 1 stream 186
Figura 6.25: Sender non RT Receiver RT – 1 stream 187
Figura 6.26: Test recenti RT + IMQ – senders 188
Figura 6.27: Test recenti RT + IMQ – receivers 189
Figura 6.28: Test recenti RT + IMQ – senders stats 190
Figura 6.29: Test recenti RT + IMQ – receivers stats 190

Capitolo 7

Ringraziamenti

Riferimenti

Appendice A

Figura A.1: Schema generale Mobile Information System 205
Figura A.2: Schema Mobile Information Server 206

Appendice B

Figura B.1: Struttura del Database nel Task 0 207

Appendice C

Figura C.1: Linux kernel packet traveling diagram 208

Figura C.2:Roadmap AEMCP protocol 209

Figura C.3: Field test sheet 210

Appendice D

Articoli

Capitolo 1

1.1 Premessa

Negli ultimi anni l'Agricoltura di Precisione (Precision Farming) e l'Automatizzazione delle Aziende Agricole (Farm Automation) in campo Agricolo richiedono strumenti e tecnologie sempre più sofisticate. In particolare si sono resi necessari sistemi di comunicazione wireless a corto raggio (short range) per la sincronizzazione delle lavorazioni e a lungo raggio (long range) per gestione flotte, e per elaborare e controllare dati di produzione in tempo reale. Sebbene alcuni strumenti e protocolli esistano già, essi soddisfano le richieste solo parzialmente. Differenti legislazioni e disponibilità di prodotti nei diversi paesi rendono molto difficile l'identificazione di una soluzione univoca per il mercato globale. In questo contesto ad esempio la suite di protocolli TCP/IP e le reti GPRS/GSM soddisfano i requisiti per una comunicazione long range, ma non quelli di una comunicazione real time necessaria per una sincronizzazione short range. Risolto il problema della comunicazione long range rimane però aperta la questione della storicizzazione dei dati, attualmente esistono differenti soluzioni proprietarie che non sono in grado di comunicare ed interfacciarsi tra loro nativamente. Ancora esistono già protocolli di comunicazione all'interno di veicoli come il CAN (Controller Area Network), ma non garantiscono sicurezza e riservatezza dei dati trasmessi. Questo documento presenta alcune proposte che mirano a fornire soluzioni, basate per quanto possibile su tecnologie esistenti, a basso costo e facilmente adottabili come standard alle problematiche sinora descritte.

1.2 Introduzione

Il lavoro presentato nasce da un 'dialogo' intercorso con ISO (International Organization for Standardization) ed in particolare con alcuni Gruppi di Lavoro ISO che si occupano di tecnologie in campo agricolo. Nel capitolo due è presente una breve introduzione ad ISO che consentirà di meglio comprendere la natura della organizzazione e quindi il dialogo con essa intercorso. Il lavoro di Dottorato si è svolto seguendo tre filoni principali, approfonditi in misura differente in funzione dell'interesse dimostrato da ISO e anche seguendo l'evolversi dello sviluppo dei lavori. In particolare le attività svolte si articolano in tre Task differenti, il primo riguarda lo studio delle possibilità di archiviazione di dati di produzione e gestione flotte tramite un sistema di database remoto, con una struttura dinamica e flessibile del

database stesso, e che fa uso esclusivamente di strumenti Open Source. Il secondo task riguarda lo studio della comunicazione intraveicolare su protocollo ISOBUS, e la proposta preliminare di alcune estensioni al protocollo che consentirebbero una trasmissione ed archiviazione sicura dei dati sfruttando tecnologie di cifratura. La terza attività infine è consistita nel progettare, realizzare parzialmente, e testare un nuovo protocollo per sincronizzazione short range che sarà oggetto di proposta per un nuovo standard internazionale. Ulteriori dettagli sono contenuti nel capitolo due.

1.3 Struttura dell'elaborato

Dopo il presente capitolo introduttivo è presente nel secondo capitolo una breve descrizione della struttura di ISO, dei rapporti con tale istituzione intercorsi che sono stati ispiratori del presente Dottorato e della suddivisione del lavoro in tre 'Task'. Nel terzo capitolo è trattato lo standard attualmente utilizzato per comunicazioni elettroniche intraveicolari in ambito agricolo, ovvero ISOBUS, insieme ad uno sguardo sul panorama attuale delle tecnologie del settore e sullo sviluppo futuro del mercato. Le problematiche connesse alla comunicazione long range in ambito agricolo e alla archiviazione e gestione di dati per fleet management e organizzazione della produzione sono oggetto del quarto capitolo, nel quale sono descritti una soluzione ideata e un sistema progettato e realizzato che implementa una proof of concept di quanto presentato. Nel quinto capitolo sono invece descritti i problemi di sicurezza nelle comunicazioni che nascono in riferimento alla soluzione proposta nel precedente capitolo, ed è delineata una possibile soluzione basata su standard già esistenti di crittografia che insieme alla estensione di alcuni protocolli già in uso permetterebbe di risolvere il problema. Una nuova proposta di standard per un protocollo realtime isocrono short range per il settore agricolo è invece oggetto del sesto capitolo, ove sono anche descritti svariati sistemi ed alcune metodologie automatiche e siti di test effettivamente realizzati. Nel settimo capitolo infine sono presentate alcune pubblicazioni e prodotti subito prima delle le conclusioni.

Capitolo 2

2.1 Introduzione - ISO

International Organization for Standardization (ISO) è la più grande organizzazione esistente al mondo che sviluppa e pubblica Standard Internazionali. ISO è in realtà costituita da una rete di istituti di standardizzazione nazionali di 162 paesi, un membro per nazione, con un Segretariato Centrale che si trova a Ginevra, in Svizzera, che coordina tutto il sistema. In Italia ad esempio il membro ufficiale di ISO è l'Ente Nazionale Italiano di Unificazione (UNI), che ha sede in Milano. ISO è una organizzazione non governativa che si propone di costituire un ponte tra il settore pubblico e privato. Da un lato, molti dei suoi istituti membri sono parte della struttura governativa nei loro paesi, oppure sono incaricati dal governo locale. D'altro canto, altri membri hanno radici unicamente nel settore privato, essendo costituiti ad esempio tramite partnership nazionali di associazioni di industriali. Pertanto, ISO consente che venga raggiunto consenso su soluzioni che soddisfino sia requisiti economici e di business che le esigenze più ampie della società civile.

2.2 ISO - cenni alla struttura

ISO è una organizzazione con una struttura estremamente complessa, esemplificata nella seguente Figura 2.1.

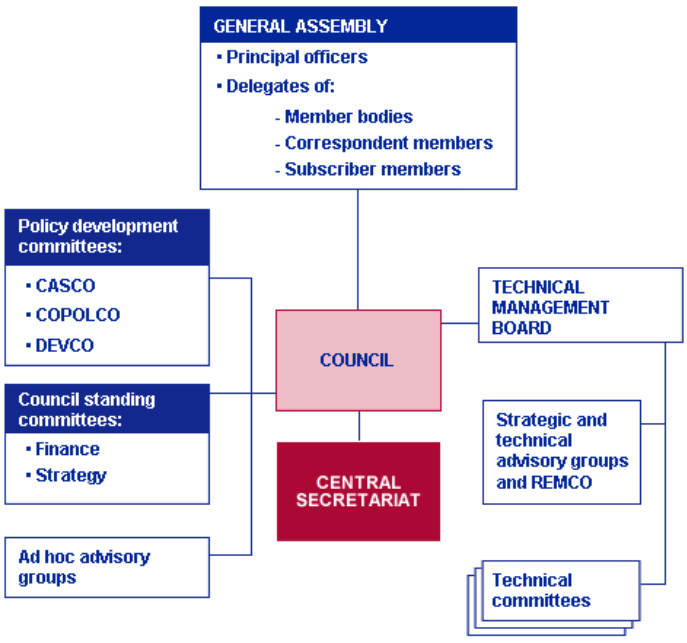


Figura 2.1: Organigramma ISO

Poiché l'interazione con ISO è stata di fondamentale importanza per l'attività di dottorato presentata in questo documento nel seguito discuteremo molto brevemente della struttura dell'organizzazione.

2.2.1 ISO - Assemblea Generale

L'Assemblea Generale è costituita dall'insieme dei dirigenti e dei delegati nominati dagli organismi membri. Soci corrispondenti e soci abbonati possono assistere come osservatori.

Come regola generale, l'Assemblea Generale si riunisce una volta all'anno. Il suo programma comprende, tra l'altro, azioni relative alla relazione annuale ISO, la compilazione di un piano strategico pluriennale con implicazioni finanziarie, e la approvazione della relazione annuale del Tesoriere sulla situazione finanziaria della Segreteria Centrale.

2.2.2 ISO - Il Consiglio

Le operazioni di ISO sono governate dal Consiglio, composto da Ufficiali e venti enti eletti dai membri.

Il Consiglio nomina il Tesoriere, i quattordici membri del Consiglio di Direzione Tecnica, e i presidenti dei comitati direttivi (policy development committees). Decide anche sul budget annuale a disposizione della Segreteria Centrale.

2.2.3 ISO - Segreteria Centrale

La Segreteria Centrale funge da segretariato per l'Assemblea Generale, il Consiglio, le commissioni della politica di sviluppo e dei loro organi sussidiari, il tecnico del consiglio di amministrazione e il comitato sui materiali di riferimento (REMCO)

2.2.4 ISO - Le Commissioni per lo Sviluppo delle Politiche

L'Assemblea Generale può istituire organismi consultivi, chiamati Commissioni per lo Sviluppo delle Politiche (Policy Development Committees). Sono aperti a tutti i membri e corrispondenti organizzazioni, e devono supportare il Consiglio nella definizione di politiche di gestione.

Attualmente esistono tre commissioni, ovvero CASCO Commissione per la valutazione della conformità (Committee on conformity assessment), COPOLCO Commissione per le politiche del consumo (Committee on consumer policy) e DEVCO Commissione per le questioni inerenti i paesi in via di sviluppo (Committee on developing country matters).

2.2.5 ISO - Le Commissioni permanenti del Consiglio

Esistono poi due commissioni permanenti, che si occupano di finanza e strategia. Il Council standing committee on finance (CSC/FIN) principalmente fornisce supporto e consulenza al Tesoriere, al Consiglio e alla Segreteria Centrale sugli aspetti finanziari delle loro attività. Il Council standing committee on strategy (CSC/STRAT) invece supporta il Consiglio nella preparazione dei piani strategici annuali, e controlla il piano strategico quinquennale di ISO. Inoltre in generale ha funzioni consultive sulle strategie da intraprendere, e riferisce al Consiglio almeno una volta all'anno.

2.2.6 ISO - Gruppi di consulenza ad hoc

Ai fini di far procedere le attività e raggiungere gli obiettivi strategici dell'organizzazione, il Presidente, con il consenso del Consiglio, può istituire gruppi consultivi ad hoc composti, tra l'altro, da figure di livello dirigenziale appartenenti ad organizzazioni aventi un interesse sostanziale alla creazione di standard internazionali. I membri di tali gruppi sono invitati dal Presidente stesso a partecipare in quanto individui piuttosto che come rappresentanti di organismi membri. Le raccomandazioni dei gruppi di consulenza ad hoc devono essere rese al Consiglio, per ogni azione successiva.

2.2.7 ISO - Consiglio Direttivo Tecnico (TMB)

I compiti del Consiglio Direttivo Tecnico, sono molteplici, di seguito ne sono evidenziati i principali:

1. Riferisce, e se necessario fornisce pareri al Consiglio circa tutte le questioni riguardanti l'organizzazione, il coordinamento, la pianificazione strategica e la programmazione di attività tecniche in ISO.
2. Esamina proposte circa nuovi campi di attività tecnica di interesse per ISO, e decide su tutte le questioni concernenti la costituzione e lo scioglimento dei comitati tecnici.

3. Verifica e mantiene in nome di ISO tutte le norme e direttive ISO/IEC, ed inoltre esamina e coordina tutte le proposte di modifica ed eventualmente approva le opportune revisioni.
4. Agisce, nel quadro delle politiche stabilite in materia di lavoro tecnico, sulle seguenti materie:
    4.1 Monitora il lavoro dei comitati tecnici ed i requisiti necessari per la gestione dei progetti in corso.
    4.2 Valuta ed approva i programmi di lavoro dei singoli comitati tecnici.
    4.3 Assegnazione o ridistribuzione delle segreterie delle commissioni tecniche e, nel caso in cui vi fosse più di un candidato, l'eventuale assegnazione o riassegnazione dei segretariati delle sottocommissioni.
    4.4 Nomina dei presidenti delle commissioni tecniche.
    4.5 Si occupa dei ricorsi contro gli interventi o le omissioni di intervento delle commissioni e sottocommissioni tecniche.
    4.6 Si occupa delle questioni di coordinamento tecnico e risoluzione dei conflitti normativi tra i comitati tecnici ISO, e nei confronti di IEC, e di altre organizzazioni nazionali ed internazionali.
    4.7 Ha funzioni consultive nei confronti della Segreteria Generale su questioni di interfaccia tecnica tra ISO e IEC, e per quanto riguarda la collaborazione tecnica con altri organismi internazionali di normalizzazione.
5. Nomina autorità di registrazione e le agenzie di gestione per l'attuazione degli Standard Internazionali.
6. Forma (e scioglie) gruppi di consulenza tecnica (TAG), al fine di ottenere consulenze di esperti di settore, e nomina i loro membri e i presidenti.
7. Forma (e scioglie) commissioni di standardizzazione su principi generali e nomina i loro presidenti.

2.2.8 ISO - Gruppi di consulenza strategica e tecnica e REMCO

I Gruppi di Consulenza Strategica (SAG) sono costituiti dal Consiglio Direttivo Tecnico (TMB) allo scopo di fornire supervisione strategica e consulenza in alcuni particolari settori, e anche per esplorare opportunità di standardizzazione in nuovi settori.

I Gruppi di Consulenza Tecnica (TAG) invece sono costituiti, quando necessario, sempre dal TMB allo scopo di coadiuvare il Consiglio Direttivo Tecnico stesso in materia di

coordinamento tecnico di base e intersettoriale, ed inoltre per consentire efficaci pianificazioni nel caso si prospettino delle nuove attività.
Le segreterie di SAG e TAG risiedono nella stessa sede della Segreteria Centrale.
Il REMCO invece è una commissione specializzata il cui compito è sviluppare Guide ISO in materia di documentazione di riferimento, con particolare attenzione alla terminologia adottata nella produzione e nell'uso di tali materiali. Esso fornisce indicazioni per le commissioni ISO nel caso in cui sia necessario fare dei riferimenti a documentazione preesistente, in pubblicazioni tecniche ISO.

2.2.9 ISO - Commissioni, Sottocommissioni Tecniche e Gruppi di Lavoro

Nella concezione di ISO, le Commissioni (TC) e le Sottocommissioni Tecniche (SC) svolgono un lavoro di gestione organizzativa, procedurale e programmatica, mentre ai Gruppi di Lavoro (WG) è demandato il compito di elaborare i progetti di norma definendone i contenuti.
Di fatto la parte pratica della normazione avviene grazie al lavoro delle commissioni, sottocommissioni e gruppi di lavoro tecnici.

2.2.10 ISO - TC23 - SC19 - WG1, WG3, WG5

Poiché le attività svolte durante il Dottorato che sono descritte nel presente documento nascono dal dialogo con alcuni Gruppi di Lavoro appartenenti ad una specifica Commissione e Sottocommissione tecnica, essi sono di seguito evidenziate. Un elenco completo delle Commissioni e Sottocommissioni Tecniche ISO può essere reperito tramite Internet, il riferimento è in Bibliografia.
Le componenti di ISO con cui si è interagito sono in particolare:
Commissione
Technical committee 23 - Tractors and machinery for agriculture and forestry
Sottocommissione
TC 23/Subcommittee 19 Agricultural electronics
Gruppi di Lavoro
TC 23/SC 19/WG 1 Mobile equipment
TC 23/SC 19/WG 3 Identification
TC 23/SC 19/WG 5 Wireless communications (sensor networks)

2.2.11 Le attività di ricerca in accordo alle richieste del WG5

L'Istituto IMAMOTER del C.N.R. è il referente Italiano per la normazione sulle macchine da lavoro e agricole e ha ricevuto mandato da UNI e CUNA per partecipare ai tavoli tecnici di scrittura delle normative in ambito elettronico, meccanico, acustico e di sicurezza, per tali tipologie di macchine. Parimenti è consulente per conto della associazione dei costruttori in seno a Confindustria, UNACOMA, per la consulenza nella interpretazione di tali normative. In quest'ambito istituzionale personale dell'Istituto fa parte dei tavoli tecnici permanenti delle commissioni e sottocommissioni e partecipa attivamente alla redazione e revisione delle normative.

Da alcuni anni è attivo il working group 5 del sottocomitato SC19 del Technical committee 23 per la redazione di normative nel campo delle reti Wireless previste per diversi tipi di utilizzo, nelle macchine agricole.

Fin dalle prime riunioni tecniche sono state enunciate le necessità principali per le quali sono previste le reti nel mondo delle macchine per le lavorazioni agricole, incentrate sul controllo di gestione delle macchine e delle farm e la programmazione del lavoro delle macchine da remoto e la sincronizzazione di più macchine che lavorino cooperativamente in una stessa area.

A fronte delle necessità sono state individuate aree di criticità per la necessità di individuare protocolli le cui frequenze fossero definite in tutto il mondo, che fossero libere in tutto il mondo e anche facilmente accessibili dal punto di vista tecnologico, possibilmente prodotte da più produttori, per garantire l'indipendenza delle soluzioni proposte.

Vista l'opportunità e le competenze a disposizione il gruppo di ricerca italiano si è fatto carico di sviluppare i protocolli di rete Wireless e fin dall'anno 2007 sono partite attività volte alla individuazione dei protocolli di rete e degli apparati necessari. In questo ambito si collocano le ricerche di questo Dottorato.

All'interno del Working Group 5 sono state definite, a partire dal novembre 2007 e a seguire nei meeting successivi, tipicamente due volte l'anno, le richieste per la definizione delle specifiche prima e delle caratteristiche dei protocolli di rete successivamente, a fronte dei risultati conseguiti durante le ricerche e presentate ai meeting e fatte circolare tra i group members circa un mese prima di ogni meeting.

Sotto la direzione del Professor Daan Goense, Project manager strategic projects at Wageningen Universtity and Researchcentre (The Netherlands) e Convener del TC23/SC19/

WG5, sono state avvallate le scelte presentate in questo lavoro; al prossimo meeting, a fine aprile 2011, saranno presentati i dati finali delle ricerche relative al protocollo WI-FI utilizzato per la sincronizzazione del lavoro cooperativo di più macchine agricole.
La complessità del campo applicativo, le diverse richieste avanzate dai produttori delle diverse aree del mondo e le effettive differenti normative applicate in Europa e in Nord America per le comunicazioni wireless hanno richiesto anni di sviluppo e aprono continuamente nuovi fronti di ricerca.

2.2.12 La nuova normativa

Il risultato più importante ottenuto durante queste ricerche nei rapporti con l'International Standard Organization, è la assegnazione della responsabilità del progetto della nuova normativa ISO per la comunicazione Wireless per le macchine agricole, la cui prima versione in stato CD (Committee Draft) è ad oggi in fase di scrittura.
Tale normativa si preannuncia complessa e articolata e già oggi ne sono delineate alcune parti che riguardano il fleet management sia dal punto di vista del protocollo che dal punto di vista della applicazione; il titolo generale della norma è già definito ed è "Tractors and machinery for agriculture and forestry — Wireless networks — Part 1: General".
La norma si articola su più parti e allo stato attuale se ne prevedono 9; è infatti prevista una Part 4 della norma che riguarda il fleet management definito nei livelli più bassi della pila ISO-OSI e una Part 2 che invece definisce i livelli più alti del protocollo, sempre seguendo la classificazione della norma ISO-OSI (ISO 7498).
Parimenti sono previste due parti di analoga struttura per la parte di sincronizzazione di macchine agricole collaborative, rispettivamente la Part 6 e la Part 5.
La Part 3 invece definisce una particolare fase della sincronizzazione delle macchine che potremmo definire come "avvicinamento" o "pre-sincronizzazione", o prendendo spunto da ambiti applicativi avionici potremmo definirla fase di "aggancio".
Infine sono in fase di definizione anche i protocolli per le Wireless Sensor Networs (WSN) e per le metodologie e i sistemi di aggancio di tali reti alle macchine agricole in campo, per permettere uno scambio dati in tempo reale al fine di determinare lo stato del campo, di maturazione dei prodotti ecc, e per raccogliere effettivamente i dati dalla rete e trasmetterli alle stazioni di terra, sfruttando le dotazioni elettroniche e informatiche previste dalle normative sulle macchine agricole e definite nelle altre parti della normativa stessa. Tali parti

sono le parti 7 e 8, che definiscono rispettivamente il protocollo a livello applicativo e a livello di definizione del protocollo per le WSN.
Infine è stata prevista una Part 9 della normativa che elenca e definisce i servizi disponibili per il trattamento dei dati raccolti in campo e dalle macchine agricole.
Per completezza è riportata sotto forma di elenco la strutturazione della normativa nelle sue parti costituenti.

Part 1 General Standards for wireless communication in agriculture.
Part 2 Long range networks, session, presentation and application layer for fleet management.
Part 3 Long range networks, session, presentation and application layer for implement synchronization.
Part 4 Long range networks, communication layer.
Part 5 Short range networks, application layer
Part 6 Short range networks, communication layer
Part 7 Ultra short range networks, session and presentation layer
Part 8 Ultra short range networks, communication layer
Part 9 Webservices for wirelessly obtained agricultural data
Le parti della normativa sotto diretta direzione e responsabilità del gruppo di lavoro italiano, sono le parti relative allo short range e al long range communication, sia per quel che concerne il protocollo, sia per quel che concerne il livello applicativo, dove peraltro è fondamentale il contributo delle aziende produttrici delle macchine, che definiscono le modalità operative relative a ciascuna lavorazione agricola collaborativa, prevedibile o attuabile già oggi sulle macchine in produzione o in sviluppo.

2.3 Approccio progettuale

La struttura complessiva del lavoro è nei successivi paragrafi riassunta ed esemplificata. Ulteriori approfondimenti sulle scelte operate, sulle tecnologie e sui componenti adottati saranno presentati nel corso dei capitoli successivi in modo da evidenziare l'approccio progettuale sin dall'inizio adottato. Nel lavoro di Dottorato infatti non si è solo inteso prospettare una soluzione tecnica al problema affrontato, ma anche tener conto che le soluzioni progettate sarebbero state oggetto di proposta per alcuni standard internazionali. Considerare anche tale aspetto significa tenere in considerazione questioni di natura non

eminentemente tecnica, ma che spesso concorrono in misura eguale se non maggiore alla determinazione dei sistemi. Per questo motivo dove necessario si è cercato di ottemperare ad esigenze spesso opposte, in una logica che deve necessariamente completare e superare la ordinaria attività del progettista di sistemi elettronici, tenendo conto ad esempio di normative internazionali, di consuetudini d'uso preesistenti nei vari paesi, di economicità, di assenza di vincoli proprietari nell'hardware e nel software, di accettabilità delle soluzioni da parte di aziende e enti normativi.

2.4 Divisione del lavoro - i tre Task

Il lavoro svolto durante il Dottorato ha seguito in particolare tre 'filoni' fondamentali, emersi considerando lo stato attuale del settore Agricolo e degli Standard già esistenti. Sono stati quindi definiti tre obiettivi fondamentali riguardanti le comunicazioni long e short range in ambito agricolo e la loro sicurezza. Nel seguito sono brevemente descritti i tre obiettivi ed i 'Task' ad essi associati. Il lavoro svolto sin dall'inizio non ha voluto essere esaustivo e conclusivo degli argomenti descritti, in quanto la compiutezza di tutte le attività chiaramente travalica gli scopi e le possibilità di realizzazione di un Dottorato di Ricerca. C'è ad esempio da considerare che tipicamente l'approvazione di uno Standard Internazionale richiede alcuni anni, travalicando quindi i limiti temporali della attuale ricerca. Cionondimeno il lavoro svolto ha costituito il tentativo di solo proporre in alcuni casi, di realizzare sistemi funzionanti o proof of concept in altri che potessero costituire una base di partenza per formulare una proposta credibile per dei nuovi Standard Internazionali in ambito agricolo. Di seguito è riportata una breve descrizione dei tre Task in cui si è articolato il lavoro complessivo.

2.4.1 Task 0

Standardizzare metodi e formati per il trasferimento di dati per comunicazioni long range in ambito agricolo (progettare un sistema 'standard' per fleet management in ambito agricolo). Di questo ambizioso progetto sono state realizzate alcune fasi, ovvero studio del contesto operativo e delle tecnologie già utilizzate e realizzazione completa di un progetto iniziale. Inoltre è stato ideato e realizzato un sistema funzionante che costituisce una Proof of Concept del progetto complessivo.

2.4.2 Task 1

Trasmissione sicura di dati attraverso un network ISOBUS ed un gateway dedicato verso Internet per comunicazioni long range per diagnosi e fleet management.
Per il Task 1 è stata effettuata tutta la parte di analisi iniziale delle tecnologie attualmente impiegate, ed è stato realizzato il progetto iniziale di una proposta di Standard Internazionale che potrebbe risolvere il problema integrando i protocolli esistenti con tecniche di crittografia, ed aggiungendo delle estensioni di semplice applicabilità ai protocolli stessi.

2.4.3 Task 2

Verificarne la fattibilità ed eventualmente progettare un protocollo compatibile con suite TCP/IP e basato su Wi-Fi per comunicazioni short range dedicate alla sincronizzazione realtime.
Sul Task 2 è stata concentrata buona della attività di ricerca, e sono state realizzate varie fasi, a partire dall'analisi del contesto attuale, proseguendo con il progetto di un nuovo protocollo di comunicazione real-time, che è stato poi implementato ed è in fase di sperimentazione. La sperimentazione ha previsto la progettazione e la realizzazione di svariati sistemi basati su piattaforme hardware differenti, la realizzazione di un laboratorio dedicato ai test, oltre a vari siti di test dislocati in sedi diverse atti a simulare varie condizioni di utilizzo. E' stato inoltre progettato ed implementato un sistema di sviluppo e test distribuito in varie sedi geografiche comunicanti per agevolare le attività ed ottimizzare l'utilizzo di servizi e risorse disponibili. Ancora è stata progettata e realizzata una serie di strumenti software che implementano la metodologia di test del protocollo in via di sviluppo.

Capitolo 3

3.1 Premessa

In questo capitolo viene presentato lo stato attuale di alcune tecnologie correntemente utilizzate nel mondo agricolo, con particolare riferimento allo standard ISOBUS. Ciò in quanto ISOBUS è la tecnologia di base per comunicazioni elettroniche a bordo di veicoli agricoli e movimento terra, ed è in pratica il fondamento a cui tutte le proposte elaborate nel nel lavoro di Dottorato hanno fatto riferimento.

3.2 Introduzione

E' comune convinzione tra gli addetti ai lavori, che l'avvento delle tecnologie legate all'elettronica nelle macchine da lavoro segua passo passo quello che avviene nel mondo automotive, mantenendo un certo ritardo, dovuto fondamentalmente alla differenza di dimensione del mercato che è considerevolmente inferiore.
Se questo è stato vero in passato, ora si può invece affermare che per certi aspetti le macchine da lavoro, e in special modo le macchine agricole, hanno preso una direzione diversa nello sviluppo di sistemi elettronici, mentre per altri aspetti sono anche più avanzate di quanto non si veda sul mercato nel mondo dei veicoli destinati al trasporto passeggeri.
Le motivazioni principali risiedono nelle sempre più accresciute necessità di precisione nelle lavorazioni e nel desiderio di tenere sotto controllo di gestione le aziende agricole così come qualsiasi altro tipo di industria o impresa.
Le maggiori innovazioni che si discostano da quello che siamo abituati a vedere nel mondo delle veicoli passeggeri sono senza dubbio i sistemi di governo e regolazione di tipo Plug & Play, ovvero universali, standardizzati dalla normativa nota come ISOBUS[1], e i sistemi di guida autonoma collegati ai sistemi informatici di programmazione delle lavorazioni nei campi.
Quello che è certamente visibile a tutti, in special modo in occasione di eventi come l'EIMA, fiera nella quale l'innovazione è presentata nelle novità tecniche e negli stand delle aziende espositrici, è che la tecnologia adottata nella realizzazione della struttura elettronica delle trattrici agricole, soprattutto di gamma media e alta, sta rapidamente cambiando, e tende ad

1. Si tratta della norma ISO 11783 sviluppata all'interno dell'ISO (International Standard Organization) e che tratta del BUS di comunicazione standard per le macchine agricole, da qui il nome ISOBUS.

adeguarsi all'ormai universalmente conosciuto standard ISO che è comunemente chiamato ISOBUS.

Le ragioni di questo adeguamento risiedono principalmente nel fatto che dietro ad una apparente maggiore complessità delle macchine e della loro struttura di controllo, si cela in realtà una maggiore semplicità se messa in relazione alle possibilità offerte da questo standard. ISOBUS, standard che descrive un protocollo di rete di tipo general purpose, per il controllo di attrezzature connesse alle trattrici - ma non solo - è descritto dalle diverse parti dello standard ISO 11783 e, a una prima lettura, appare come un protocollo strutturato che si basa sulla realizzazione di una rete di tipo CAN High Speed, che permette l'accoppiamento della trattrice con gli attrezzi a essa collegati, non solo dal punto di vista meccanico, idraulico ed elettrico, ma anche elettronico e di controllo.

Appariva infatti come una lacuna normativa la mancanza di uno standard in un campo di applicazioni dove tutto il resto era stato standardizzato sia dal punto di vista meccanico(PTO e Hitch) che idraulico (attacchi rapidi) ed elettrico:

La mancanza di uno standard per il controllo remoto non solo della potenza meccanica, idraulica ed elettrica fornita agli attrezzi è stata quindi colmata da questo documento.

3.3 Normazione e organismi transnazionali

La standardizzazione dei sistemi di controllo elettronico distribuito nelle applicazioni agricole rappresenta un intero mondo, al punto che oltre a un nutrito gruppo di esperti nazionali, provenienti dalle più importanti aziende di settore, che si occupano di concepire e scrivere la normativa, sono nate associazioni per lo studio e per la creazione di linee guida e strumenti per favorire l'adozione dello standard da parte delle aziende. Tali organismi erano l'NAIITF (North American ISOBUS Implementation Task Force) e l'IGI (ISOBUS Group Implementation) patrocinato da VDMA e DLG che avevano messo in campo un gruppo di esperti per il supporto alle aziende nordamericane ed Europee rispettivamente. Ora tali organismi, in accordo alle proprie Associazioni nazionali (rispettivamente l'AEM e il VDMA) si sono sciolte dando luogo ad un unico organismo mondiale che prende il nome di "AEF e. V." Agricultural Industry Electronics Foundation Eingetragener Verein, una forma societaria tedesca e austriaca di associazione di imprese. AEF è fondata dalle maggiori aziende di settore e tra i membri fondatori si possono trovare tutte le maggiori aziende mondiali di produzione di

trattrici agricole. La forte connotazione applicativa di AEF indica la volontà di sviluppo e implementazione della norma da parte delle aziende.
L'analisi delle motivazioni di tale spinta innovativa è scuramente complessa, ma uno dei maggiori motivi che hanno generato la collaborazione di aziende - che sono a tutti gli effetti in concorrenza diretta - è la necessità di analisi profonda delle implicazioni di controllo e di sicurezza che riguardano il pilotaggio di attrezzi attraverso un sistema elettronico distribuito che a tutti gli effetti rappresenta un sistema di tipo by-wire sul quale va concentrata la massima attenzione possibile, per la realizzazione di macchine funzionali e sicure.

3.4 Che cosa è ISOBUS

ISOBUS è uno standard internazionale che definisce in modo completo un sistema di unità elettroniche in grado di creare un sistema tale da permettere la interconnessione di qualsiasi attrezzo a qualsiasi trattrice agricola, utilizzando sempre gli stessi sistemi di comando e controllo, senza la necessità di sostituire i comandi in cabina ogni volta che venga sostituito l'attrezzo collegato alla trattrice.
Il sistema normato prevede l'utilizzo di attrezzi controllati attraverso sistemi elettronici che si scambiano messaggi attraverso una rete locale che attraversa la trattrice e prosegue sull'attrezzo, fino a raggiungere tutte le centraline di controllo dislocate in esso, direttamente, o attraverso dei gateway che dividono la rete in sezioni.
I sistemi elettronici che governano il sistema sono molti e ciascuno di essi è specializzato in una particolare funzione che, associata alle altre, permette l'esecuzione di tutte le funzioni necessarie al corretto utilizzo della trattrice e degli attrezzi ad essa collegati.
Il cuore funzionale di tale sistema è costituito dall'idea forte che lo ha fatto nascere, ovvero la identificazione automatica della tipologia di attrezzo e delle sue modalità di governo, regolazione e controllo, attraverso una comunicazione di tali caratteristiche in rete, che si esplicitano in una serie di informazioni visibili per l'operatore in trattrice su uno speciale monitor - detto Virtual Terminal - in grado di visualizzare i dati dell'attrezzo in modalità grafica e di configurare qualsiasi attrezzo.
A tutti gli effetti ISOBUS è quindi un sistema di tipo Plug & Play, e per spiegarne tale caratteristica si potrebbe paragonare la connessione di un attrezzo agricolo alla trattrice a quella di una qualsiasi periferica USB ad un PC.

ISOBUS in realtà è molto più di questo, perché dal suo anno di nascita (1993) è diventato un vero e proprio sistema di gestione delle lavorazioni agricole eseguite con una trattrice e con un set di attrezzi, sotto il completo controllo dei sistemi elettronici e dei sistemi informatici di programmazione delle lavorazioni residenti nei computer della azienda agricola.
Se infatti il cuore del sistema è costituito dalla stretta comunicazione tra attrezzo e sistemi di comando in trattrice (leve, joystick, pulsantiere e ovviamente il Virtual terminal), il sistema in realtà è dotato di una serie di unità speciali che permettono la gestione completa della sessione di lavoro, a partire dalla sua programmazione e registrazione di tutti i dati sensibili fino ad arrivare alla guida autonoma della trattrice su base di informazioni GPS ad alta definizione.
Non si parla di futuro o di prototipi, ma di sistemi in produzione di serie sulle trattrici di gamma alta e media delle principali aziende del settore.

3.5 ISOBUS: La struttura della rete

Una esposizione generale delle caratteristiche di tale sistema di gestione delle macchine agricole a controllo distribuito non può prescindere dalla descrizione delle principali unità di controllo in esso presenti. Tutto il funzionamento e la gestione del sistema è regolato da un passaggio di informazioni attraverso due reti i tipo CAN, la rete a bus di campo di uso automotive per eccellenza, introdotta da Bosch nei primi anni '90.
La struttura di base della macchina agricola dotata di rete ISOBUS è costituita da due reti di tipo CAN:
1.la prima è la classica rete Powertrain aderente alla normativa SAE J 1939, che è presente nelle trattrici agricole fin dal momento in cui è stato necessario adottare motori diesel a controllo elettronico per la riduzione delle emissioni inquinanti, come è avvenuto nel mondo dei veicoli stradali Heavy Duty; a tale rete afferiscono le unità responsabili del funzionamento della parte di controllo della trattrice: motore , cambio, controllo delle sospensioni, dei sistemi di frenatura e sterzatura elettronica ecc.
2.La seconda invece è la vera rete ISOBUS, ovvero la rete che permette il controllo remoto degli attrezzi attraverso il pilotaggio by-wire mediante messaggi che circolano sulla rete CAN.
I principali componenti della rete ISOBUS sono i sistemi di setup della rete e dell'attrezzo, i comandi - normalmente residenti nella cabina della trattrice - , i sistemi di comando e controllo - che possono essere residenti sia in trattrice che nell'attrezzo - e gli attuatori - normalmente residenti nell'attrezzo Figura 3.1.

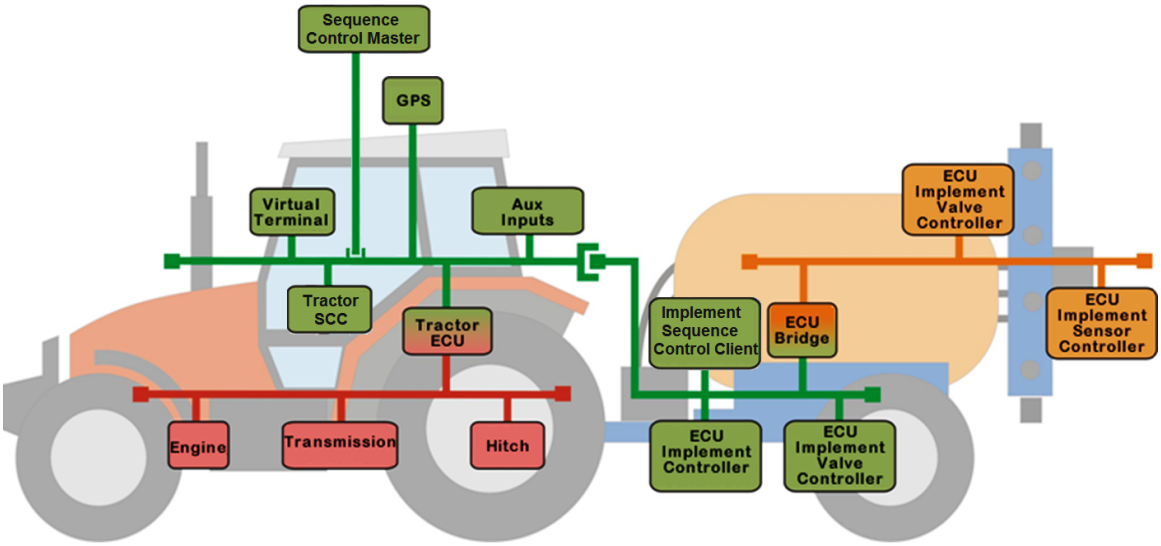


Figura 3.1: Struttura della rete ISOBUS e della rete Powertrain con le principali unità elettroniche di controllo

3.6 Il Virtual Terminal

Per coloro i quali non si siano mai soffermati a osservare il funzionamento di questa tipologia di rete può non essere chiara la grande novità di ISOBUS, che è anche la sua maggiore potenzialità: ISOBUS è a tutti gli effetti un sistema di controllo e comando di qualsiasi tipologia di dispositivo remoto in rete CAN. Qualsiasi attrezzo o dispositivo che soddisfi i requisiti della normativa può collegarsi alla rete a presentare i propri "metodi" di comando e controllo al dispositivo di comunicazione e scambio di informazioni con l'operatore: il Virtual Terminal, un display dotato di tasti funzione riconfigurabili, che presenta tante pagine grafiche quante sono quelle richieste dinamicamente dagli attrezzi che in quel momento sono stati collegati alla trattrice. Un esempio di Virtual Terminal è visibile in Figura 3.2. Il Virtual terminal nato come sistema di interfaccia con gli attrezzi è ormai diventato molto di più. La struttura delle reti della trattrice e logiche di integrazione e di efficienza nella gestione dei dati, hanno fatto si che nel Virtual Terminal venissero raccolte anche altre funzionalità; pur variando da casa costruttrice a casa costruttrice le caratteristiche aggiuntive dei Virtual Terminal spesso si ritrovano nelle trattrici di diversi produttori: prima tra tutte la serie di pagine grafiche che presentano i classici dati motore e di marcia del veicolo, ovvero i dati raccolti dalla rete Powertrain, perché spesso nel VT vengono portate direttamente entrambe le reti, al fine di visualizzare quante più informazioni possibile, indipendentemente da quante informazioni transitino effettivamente da una rete all'altra per motivi legati ai controlli degli attrezzi e delle lavorazioni.

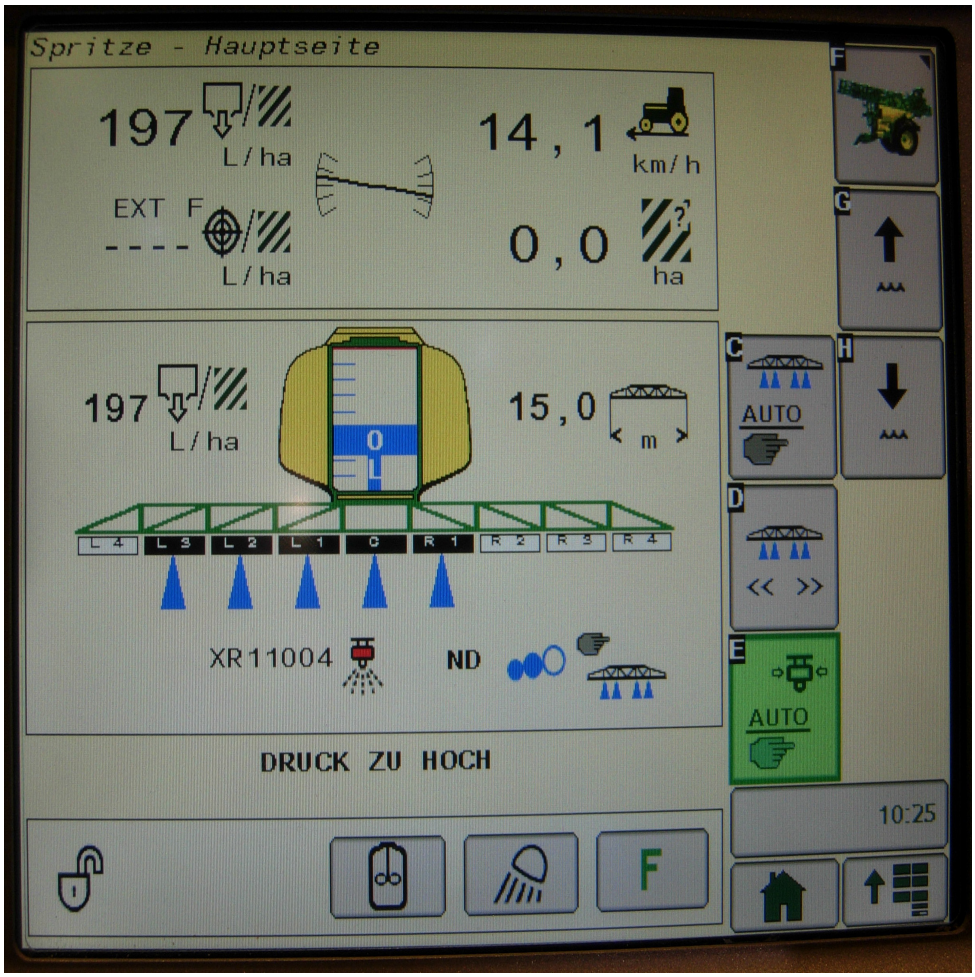


Figura 3.2: Virtual Terminal John Deere screenshot schermata di comando e monitoraggio di Sprayer

Viceversa vi sono alcune funzionalità accessorie che devono risiedere nel VT, quali ad esempio le pagine di configurazione dei comandi riconfigurabili - detti Auxiliary Input - a disposizione dell'operatore, perché tale Terminale è l'unica interfaccia attraverso la quale l'operatore può effettivamente intervenire sul funzionamento di tutti i dispositivi di comando configurabili presenti nella cabina della trattrice.

3.7 Gli Auxiliary Input

Attraverso il Virtual Terminal è infatti possibile configurare un attrezzo e comandarlo, sia attraverso comandi impartiti direttamente dal terminale, sia configurando l'uso di interfacce di comando quali joystick, pulsanti, roller e leve presenti sulla trattrice comunemente detti Auxiliary Inputs. Anche queste serie di dispositivi di comando sono componenti a norma

ISOBUS e sono dispositivi configurabili, associabili in modo completamente libero alle funzionalità dei diversi Implement che di volta in volta siano collegati alla trattrice.

Figura 3.3: Consolle di comando FENDT con Auxiliary Input e relativa pagina di configurazione sul Virtual Terminal

Tali Interfacce di comando azionabili dall'operatore, non sono altro che dispositivi di comando il cui azionamento genera un messaggio che è pubblicato sulla rete e che viene recepito dalla unità di controllo elettronico, che è stata associata a quel comando e che deve esplicitare l'azione di controllo corrispondente (Figura 3.3).

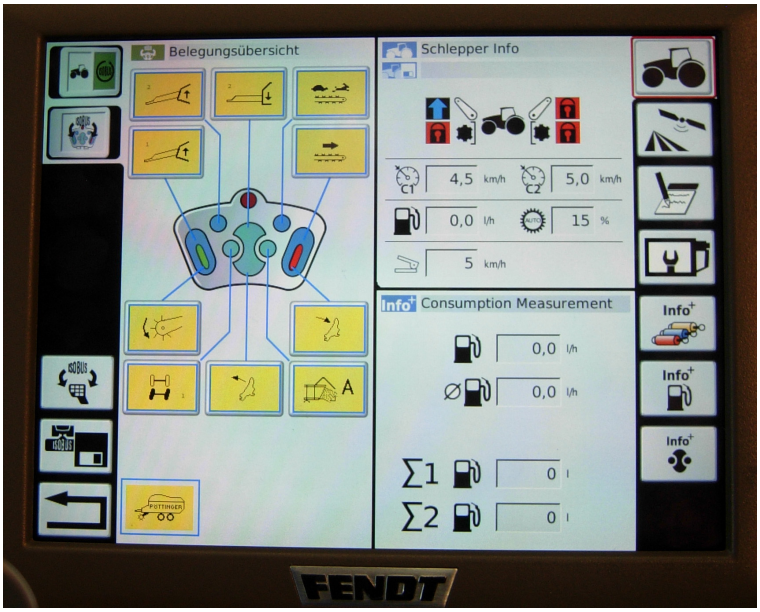


Figura 3.4: Pagina di associazione tra Auxiliary Input disponibili e funzionalità comandabili da remoto in una trattrice FENDT

Si ha quindi una dislocazione remota dei comandi in rete attraverso l'azionamento di dispositivi di comando azionabili manualmente, completamente configurabili attraverso una serie di pagine speciali residenti nel Virtual Terminal. In particolare esistono pagine nelle quali si può realizzare tale associazione, simili a quella mostrata in Figura 3.4 dove nella colonna in giallo sono visibili le funzionalità della trattrice azionabili da remoto, e pagine di controllo e verifica globale dei settaggi realizzati, quale quella mostrata in Figura 3.5 che ha lo scopo di ricordare all'operatore i settaggi per evitare errori di comando, appare ad ogni accensione del sistema ed è richiamabile anche durante la sessione di lavoro:

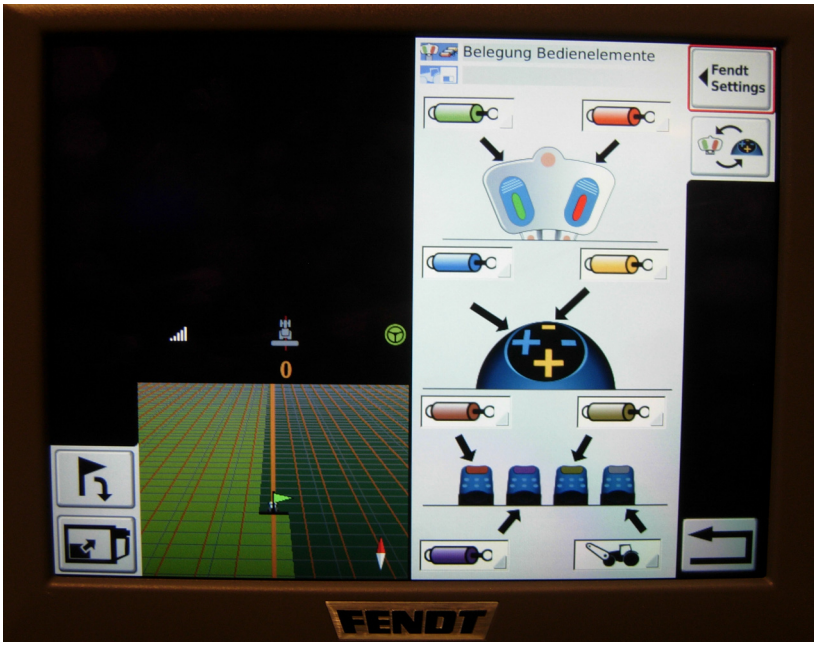


Figura 3.5: Pagina di richiamo delle associazione realizzate tra Auxiliary Input disponibili e funzionalità comandabili da remoto in una trattrice FENDT

Tale struttura costituisce un vero e proprio comando di tipo by-wire, perché pilota attuatori di tipo diverso attraverso messaggi inviati in rete CAN, nel caso particolare utilizzando il protocollo ISOBUS come standard per la trasmissione del comando.
In altri tipi di veicoli un sistema di questo tipo può utilizzare protocolli di tipo diverso, ad esempio CANOpen, ma la sostanza non cambia: si tratta di un sistema di pilotaggio di tipo actuation by-wire, nel quale una attuazione comandata direttamente è sostituita da un sistema di unità di controllo elettronico distribuite in rete e dislocate in punti diversi del veicolo, che si occupano di interpretare un comando e passarne il valore interpretato alla unità responsabile della attuazione.

Tale struttura, al di la delle problematiche di sicurezza che inevitabilmente devono essere affrontate, offre una grande versatilità e anche una semplificazione dei sistemi più complessi, oltre all'ovvio vantaggio immediato di non dover installare niente in cabina, a tutto guadagno della ergonomia, della sicurezza, della rapidità di installazione di un attrezzo e anche del colpo d'occhio.

3.8 La TECU

Oltre alle unità funzionali all'effettivo comando degli attrezzi, il protocollo ISOBUS prevede una serie di unità di controllo asservite al corretto funzionamento dell'intero sistema, con compiti legati alla sicurezza di marcia e alla supervisione delle richieste dell'operatore e degli attrezzi, che, in base al tipo di lavorazione in atto, possono essere avallate o rifiutate, a vantaggio della sicurezza del veicolo, della lavorazione e dell'operatore. La più importante di tali unità è la Tractor ECU (Electronic Control Unit), detta TECU, che è una unità preposta alla regolazione del traffico di rete tra la rete Powertrain e la rete attrezzi, nonché alla supervisione delle richieste che provengono dagli attrezzi e dai sistemi di guida autonoma e sono dirette alle componenti Powertrain della trattrice, con notevoli effetti sulla sicurezza. Infatti le richieste possono riguardare sia richiesta di potenza e coppia, necessarie ad un attrezzo per svolgere un determinato lavoro, oppure richieste di velocità di rotazione della Presa Di Forza, oppure ancora richieste di posizionamento dell'attacco a tre punti, per la regolazione della posizione di un attrezzo. Ma in modo più rivolto al moto, possono essere richieste da parte degli attrezzi o di dispositivi di guida autonoma anche velocità e potenza del veicolo e un determinato numero di giri del motore diesel. Tutto questo senza che l'operatore debba in qualche modo intervenire, se non per dare una sola iniziale abilitazione a queste funzioni di marcia automatica. Inutile dire che questa tipologia di comandi remoti ha delle implicazioni di sicurezza non banali e proprio per questo alla TECU sono demandati anche compiti di supervisione per l'effettivo avvallo di queste richieste, che sono lasciate transitare dalla rete attrezzi alla rete powertrain solo se le condizioni di marcia e la configurazione del veicolo in lavoro lo consentono.

3.9 Il Task Controller e il File Server

Tra le unità che possono intervenire attivamente, sia sulla marcia della trattrice, sia sulla attivazione delle funzioni degli attrezzi a controllo elettronico aderenti a ISOBUS, c'è il Task Controller, probabilmente la più complessa unità di tutto il sistema. Tale unità di controllo è in grado di associare ad un veicolo e agli attrezzi ad esso connessi, un programma di lavoro, identificato sia a una area geografica in cui tale lavoro deve essere svolto, sia da una serie di grandezze che identificano in modo completo la tipologia di lavorazione da eseguire in quell'area. Nel Task Controller sono definite tutte le grandezze relative ai prodotti da lavorare con gli attrezzi, sia la struttura, le dimensioni e le funzionalità automatizzabili degli attrezzi a disposizione, in modo da permettere la completa gestione automatica dell'attrezzo una volta che la trattrice si trovi nell'area predeterminata per la sessione di lavoro.
I dati relativi alla lavorazione sono inseribili sia attraverso il Virtual Terminal che attraverso sistemi proprietari costituiti da applicazioni su PC.
Il funzionamento del Task Controller si basa sulla generazione di comandi, del tutto simili a quelli che avrebbe impartito l'operatore, agli attrezzi in corrispondenza dei punti GPS in cui le lavorazioni devono essere eseguite.
Come si può vedere in Figura 3.6

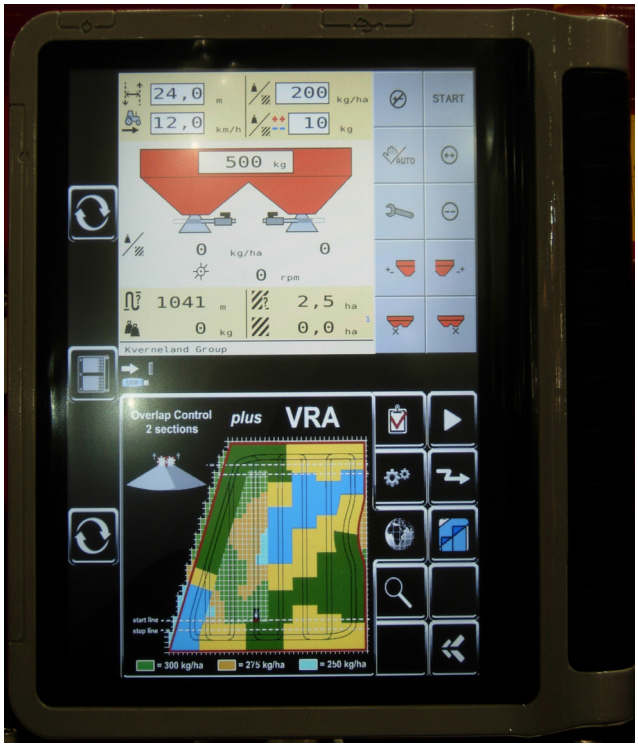


Figura 3.6: Virtual terminal Kverneland con le due aree relative al controllo dell'attrezzo (alto) e al controllo di lavorazione mediante Task Controller (basso).

e Figura 3.7,

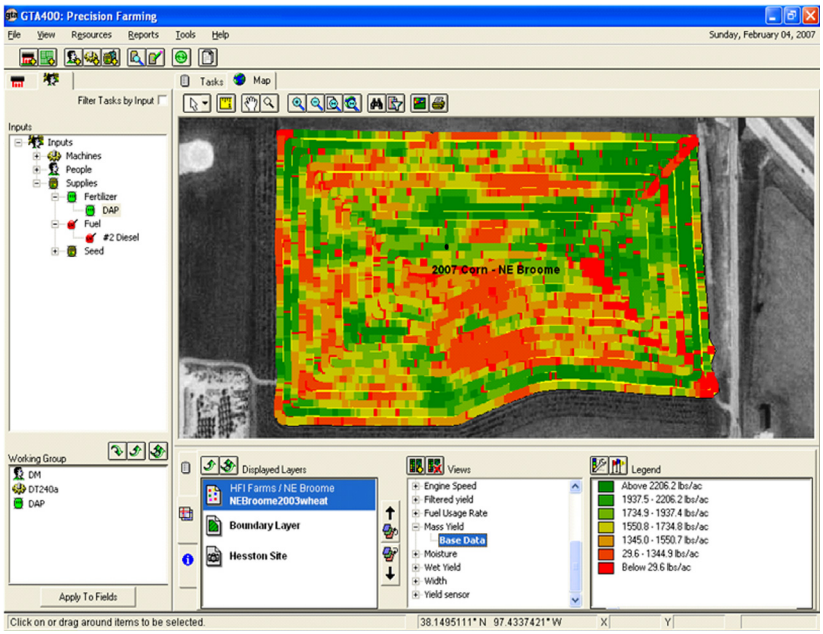


Figura 3.7: Applicazione su PC per la programmazione delle lavorazioni su task Controller ISOBUS e per visualizzazione dei dati da File Server

I dati si basano sostanzialmente su una immagine del campo e sul tipo di lavorazione da eseguire. Ad ogni punto GPS sarà quindi associato una configurazione degli attrezzi e una quantità di prodotto. Se ad esempio si considera uno sprayer, si avrà un dato relativo alla quantità di prodotto da erogare attraverso i nozze posizionati sui bracci (boom) dello sprayer, per ogni punto del campo. Da qui gli sviluppi sono notevoli: c'è chi associa a programmi di gestione delle lavorazioni le immagini aree prese attraverso dei piccoli veicoli UAV (Unmanned Aerial Vehicle) elaborate in tempo reale, che permettono, attraverso indagini colorimetriche e altro, di determinare la quantità esatta di prodotto da erogare.

Perché comunque sia possibile la gestione automatizzata della lavorazione gli attrezzi devono poter accettare comandi automatici da un Task Controller, così come per poter gestire un sistema di guida automatizzato, la trattrice deve essere anche dotata di un sistema GPS ad

elevata precisione (come minimo un sistema WAAS/EGNOS[1] o un DGPS[2]) e di un sistema di guida automatico.

Per chiudere il processo di controllo di gestione della lavorazione è importante anche la registrazione dei dati effettivi della sessione portata a compimento dalla macchina e a tale scopo è prevista una unità per la registrazione di tutti i dati sensibili del lavoro svolto. Si tratta di una unità di data logging capace di gestire un file system in stile PC e in grado di "sniffare" dalla rete ISOBUS tutti i messaggi contenenti dati importanti per la definizione della posizione, della marcia del veicolo e del funzionamento istantaneo degli attrezzi. Grazie a questo tipo di informazioni è possibile ottenere grafici come quello di Figura 3.7, che consentono di calcolare con precisione la resa per ettaro delle coltivazioni, al netto delle reali spese di gestione.

3.10 Il Sequence Controller

Sempre nell'ottica di ridurre il carico di lavoro degli operatori, e, a nostro parere, di rendere il più possibile le operazioni indipendenti dal particolare operatore, ISOBUS prevede una ulteriore unità di controllo, che spesso è integrata con la precedente: il Sequence Controller.
Tale dispositivo permette all'operatore di registrare una serie di operazioni da eseguire in sequenza, che verranno ripetute in modo identico molte volte durante la sessione di lavoro, e di farle ripetere in modo automatizzato al sistema elettronico grazie alla ripetizione dei comandi impartiti alle diverse parti della macchina e agli attrezzi nello stesso ordine e con le stesse modalità. Le tipiche operazioni di questo tipo sono le operazioni dette "di bordo campo"; tipicamente arrivati alla fine del campo coltivato si deve invertire la marcia del mezzo fermando momentaneamente l'attrezzo o sollevandolo (si pensi ad un aratro), diminuendo contemporaneamente la velocità della trattrice per permettere l'inversione del senso di marcia. Tutte queste operazioni possono essere eseguite un'unica volta e poi ripetute in modo completamente automatizzato o mediante un unico comando impartito dall'operatore,

1. WAAS/EGNOS: Wide Area Augmentation System / European Geostationary Navigation Overlay Service sono due Sistemi di increment della precision dei Sistemi di posizionamento satellitare, il primo di concezione Americana e il secondo europeo, che si basano sulla triangolazione di satellite geostazionari e di antenne fisse sul suolo terrestre che trasmettono i dati di correzione delle rilevazioni satellitari, dato che la loro posizione sulla terra è fissa e nota.
2. DGPS: Differential Global Positioning System, sistema di posizionamento satellitare che si serve anche di stazioni locali di terra per una riduzione dell'errore dei segnali dei satelliti e permette, in taluni casi, di ridurre l'errore di calcolo della posizione di soli 2 - 3 cm.

oppure sfruttando un "geofence[1]" controllato grazie al sistema GPS, ovvero la condizione di avvicinamento al bordo del campo, ovvero al termine dell'area definita come "area di lavoro" per quella sessione.
In Figura 3.8

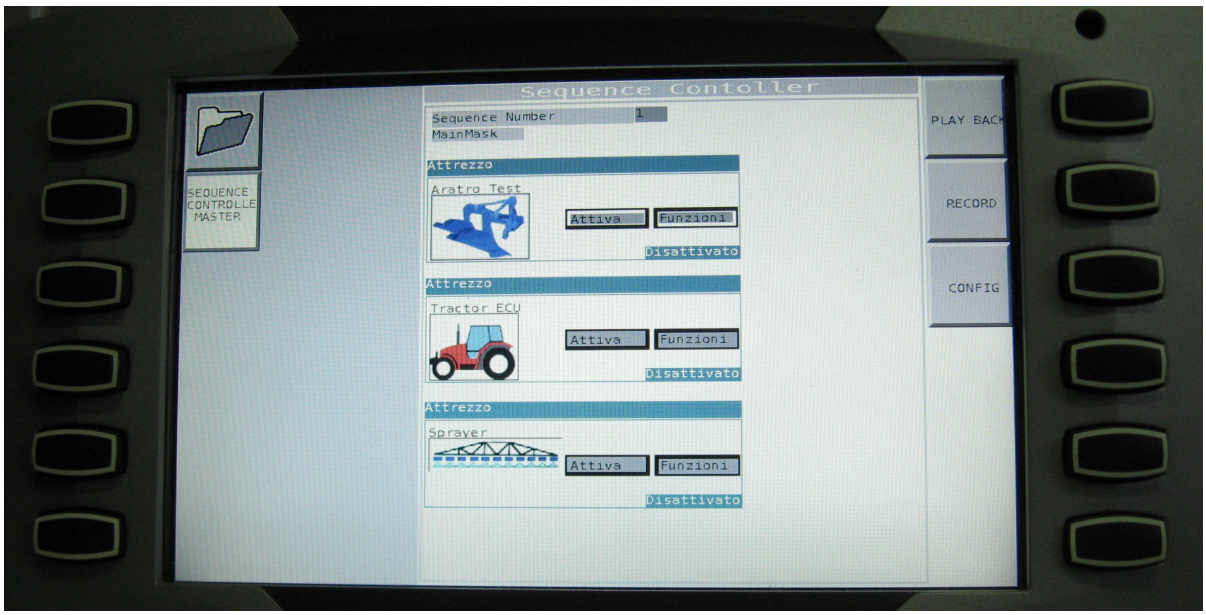


Figura 3.8: Applicazione su PC per la programmazione delle lavorazioni su task Controller ISOBUS e per visualizzazione dei dati da File Server

è presentato un esempio di sistema di Sequence Controller gestito attraverso il Virtual Terminal, si possono notare i simboli degli attrezzi che permettono una gestione automatizzata dei comandi mediante il Sequence Controller. È importante notare che il Sequence Controller è in grado, al pari del task Controller, di eseguire manovre di guida autonoma, tenendo conto della larghezza dell'attrezzo, e quindi dell'area di campo già lavorata o da lavorare, come si può notare a esempio in, dove è chiaramente indicata l'area di campo in cui la trattrice è già passata e l'area ancora da trattare, e le aree in cui, per necessità di manovra, si è passati più di una volta; in tali aree si ha la possibilità di fermare l'attrezzo anche in modo selettivo (gruppi di ugelli in uno sprayer ad esempio), per evitare di eseguire lavorazioni errate o eccessive.

1. Steccato virtuale: un'area definita entro la quale la macchina deve operare e della quale è in grado di riconoscere i confini e la distanza da essi.

3.11 I sistemi di guida autonoma

Vale la pena infine citare i sistemi di guida autonoma, che esistono indipendentemente da ISOBUS ma che possono essere integrati in esso, ereditando tutti i vantaggi della lavorazione completamente automatizzata delle lavorazioni nei campi. Ormai esistono in commercio alcuni modelli, più o meno integrati nella trattrice; molti sono sistemi che si aggiungono al sistema di guida manuale adattando un comando sul piantone dello sterzo. Tutti questi sistemi sono concepiti per essere sicuri e sono normalmente dotati di ridondanza nella elettronica di controllo, in modo da assicurare la corretta gestione della guida, ma sono anche meccanicamente concepiti per non essere pericolosi, per cui sono sempre dotati di sistemi di sblocco, basati normalmente sulla forza erogata dal sistema elettronico sullo sterzo, che permette in ogni istante all'operatore di riprendere il controllo della guida escludendo il sistema elettronico.
Va detto che in base ad alcune campagne di test eseguite in ambiti "standard", ovvero in campi privi di problematiche particolari, si può affermare che l'operatore è necessario sulla trattrice solo per obblighi normativi, dato che l'intera sessione di lavoro è stata svolta in completa autonomia dalla macchina governata dalle unità di controllo ISOBUS.
Diverso, ad oggi, il caso in cui si abbiano campi in terreno scosceso o con aree non agibili o comunque pericolose (argini, pendii ripidi, terrazze ecc.). Ma le maggiori aziende del settore stanno lavorando molto intensamente sui sistemi di guida, che rappresentano la sfida tecnologica di oggi e un passo importante per la affermazione sul mercato degli interi sistemi di gestione delle macchine, per cui si prevede che presto i sistemi saranno completi e affidabili su qualsiasi terreno.

3.12 La progettazione dei componenti e le aziende coinvolte

Dal punto di vista degli sviluppatori, ISOBUS rappresenta una soglia non facilmente superabile per i progettisti di sistemi elettronici per macchinari agricoli: la grande quantità di codice necessaria per la implementazione del protocollo, la dimensione dei database dei comandi e gli strumenti necessari, oltre alla necessità di potenza di calcolo e della presenza di sistemi operativi real time nelle unità di controllo, rendono impegnativa la realizzazione di un sistema aderente allo standard; il prevedibile risultato è che ISOBUS contribuirà a realizzare una selezione tra le aziende che operano come fornitrici di sistemi elettronici per il mondo delle macchine agricole, riducendo il numero delle aziende in grado di fornire

componentistica certificata e di fatto aumentando la qualità media dei prodotti disponibili. Per le aziende coinvolte c'è comunque sia la possibilità di realizzare l'intero sistema al proprio interno, cosa che potrebbe impegnare per un paio di anni prima di veder nascere un prototipo funzionante e aderente allo standard, sia la possibilità di acquistare il protocollo standard da aziende specializzate che lo commercializzano, risparmiando tempo e tutti i costi relativi alla certificazione del protocollo, concentrandosi solo sulla realizzazione della applicazione, che costituisce probabilmente il core business e la vera competenza della azienda.

3.13 L'evoluzione dello standard per i prossimi anni

ISOBUS rappresenta uno standard ormai maturo, le parti fondamentali ella norma subiscono variazioni davvero minime ad ogni revisione, e le modifiche maggiori rappresentano ad oggi adeguamenti alle nuove normative sulla sicurezza dei sistemi elettronici. Tuttavia, con l'innovazione delle tecnologie, si hanno nuovi orizzonti di sviluppo anche per lo standard ISOBUS; una delle maggiori innovazioni è rappresentata dal controllo remoto su rete cellulare GPRS/UMTS, che si fa strada anche per le flotte di veicoli: esistono molte applicazioni cosiddette di Fleet Management sul mercato e attualmente il gruppo di lavoro ISO per la normazione della elettronica per le macchine agricole sta standardizzando proprio questo tipo di applicazione. Ma questa non è l'innovazione più importante, che invece è rappresentata dalle ricerche che lo stesso gruppo di lavoro sta portando avanti per lo sviluppo di un sistema di comunicazione a corto raggio che permetta il lavoro sincronizzato dei veicoli; il caso più rappresentativo del tipo di applicazione che si può ottenere è rappresentato dal trasferimento del carico da una mietitrebbia ad un camion o a una trattrice dotata di rimorchio, durante la lavorazione di trebbiatura, senza che i veicoli siano costretti a fermarsi e senza che sussistano rischi di collisione (Figura 3.9). I due veicoli sono tra di loro collegati mediante una rete Wireless, e sono in grado di seguire un cammino parallelo, basandosi sulla propria posizione GPS, sulla velocità e sulla direzione che è mantenuta costantemente uguale tra i due veicoli, uno dei quali è definito master e l'altro, che segue, slave nel processo di caricamento.
La difficoltà principale di tale standard è garantire la periodicità del trasferimento di dati, per evitare che il ritardo nella comunicazione della direzione e della velocità da parte del master , faccia deviare lo slave e provochi dei danni.

.

Figura 3.9: Marcia parallela di Harvester e trattrice con caricamento di materiale durante la marcia dei veicoli.

Per questo tipo di applicazione forse si dovrà aspettare ancora un anno o due, ma i primi test sul campo sono già stati effettuati e i problemi tecnologici maggiori sono stati risolti. In ogni caso si può affermare che il lavoro cooperativo di veicoli autonomi rappresenta uno dei campi di ricerca applicata nel campo agricolo che sta avendo il maggior incremento in tutti i paesi maggiori produttori di macchine agricole.

3.14 La previsione di mercato per l'immediato futuro

Agli occhi di chi legge potrà apparire eccessivo un sistema di tale genere, soprattutto se rapportato alla dimensione di una tipica azienda agricola a conduzione familiare italiana; non si dimentichi però che in altri paesi l'estensione media delle colture è maggiore, e che quando si parla di colture massive si parla di centinaia di ettari. In stati come il Canada, gli Stati Uniti, la Germania e l'America Latina, i sistemi di guida e ISOBUS sono ormai uno strumento fondamentale. ISOBUS è ormai una realtà ed è prevedibile un incremento del mercato degli attrezzi ISOBUS in virtù della presenza di componentistica ISOBUS su tutte le trattrici di

fascia alta, pur nella consapevolezza che per anni continueranno a coesistere le versioni non-ISOBUS degli attrezzi.

Ma la estrema comodità dei sistemi automatizzati, la semplicità di uso e di connessione è probabile che invoglino gli utilizzatori - soprattutto i più giovani, abituati orami all'uso di PC e consolle di gioco - ad intraprendere la via di ISOBUS, da cui difficilmente torneranno indietro.

Capitolo 4

4.1 Task 0

In questi ultimi anni l'Agricoltura di Precisione (Precision Farming) e l'Automazione delle Aziende Agricole (Farm Automation), rendono sempre più urgente la necessità di comunicazioni wireless a lunga distanza (long range) per gestione flotte, e per controllo e organizzazione della produzione in tempo reale e differito. L'utilizzo di tali strumenti infatti consentirebbe di aumentare in modo significativo l'efficienza della attività agricola con evidenti vantaggi. Ad esempio un utilizzo efficace dei fertilizzanti che tenga conto delle caratteristiche del terreno, oppure dati di produzione elaborati in tempo quasi reale, e trasmessi alla intera filiera di gestione dei prodotti agricoli rendono possibile l'ottimizzazione dei processi produttivi minimizzando sprechi e scarti. Esistono già alcuni Standard Internazionali nel campo delle comunicazioni cablate intraveicolari, come ad esempio ISOBUS descritto nel capitolo tre, ma tali Standard soddisfano i requisiti solo parzialmente, non consentono comunicazioni wireless, e non prevedono modalità di immagazzinamento e gestione dei dati. A ciò si aggiunge il fatto che legislazioni e disponibilità di prodotti differenti nei diversi paesi rendono complessa l'identificazione di una soluzione per il mercato globale. Il Task 0 ha riguardato lo studio e la progettazione di una architettura di sistema di gestione e immagazzinamento dati basata quanto più possibile su strumenti esistenti. Tale architettura vuole rappresentare una proposta complessiva per la soluzione del problema della gestione e immagazzinamento dati in ambito agricolo che sia adottabile come Standard mondiale. Come compendio alla parte di ideazione della architettura è stata poi progettato e realizzato un sistema funzionante a titolo di proof of concept. Anche tale sistema viene brevemente descritto nel seguito.

4.2 Task 0 - Contesto attuale

In realtà esistono attualmente alcuni sistemi che consentono monitoraggio e amministrazione di una flotta di veicolo agricoli e dei dati di produzione tramite l'installazione di terminali mobili sulle unità. Però le soluzioni esistenti sono caratterizzate da svariati problemi che di fatto le allontanano dalla adottabilità quali Standard Internazionali. Ad esempio gli applicativi per la gestione dei terminali sono caratterizzati da notevole rigidità, e tutti i software esistenti sono attualmente proprietari e quindi legati al particolare tipo di terminale adottato. Dunque

nel caso di adozione di terminali prodotti da aziende diverse se anche fosse possibile una gestione comune, essa andrebbe effettuata utilizzando separatamente tecnologie diverse oppure integrandole caso per caso. Ancora spesso la scelta del terminale da adottare è vincolata perché i terminali non sono generici, ma in grado di effettuare solo alcune tipologie di rilevamento e trasmissione dati, e non tutti sono disponibili nei vari mercati mondiali, anche semplicemente per motivi commerciali. E' evidente come tale situazione sia ben lontana dall'essere ideale.

4.3 Finalità progettuali

Nella gestione di una flotta capita assai di frequente di voler conoscere lo stato corrente delle unità che la compongono, siano esse automobili, mezzi pesanti, imbarcazioni o mezzi si altra natura. Nel caso dei mezzi agricoli si è già accennato a quali grandi vantaggi sulla produttività la pronta disponibilità di dati porterebbe.

Per ottenere i dati necessari è quindi necessario immaginare una rete di dispositivi 'terminali' installata a bordo dei veicoli della flotta, ed in grado di comunicare ad una certa distanza, che realizzano una rete ed in grado di fornire dati di vario genere.

Tali dati possono essere di tipologie molto diverse, devono viaggiare bidirezionalmente ed essere opportunamente immagazzinati in sistemi in grado di memorizzarne e gestirne agevolmente anche grandi quantità, e che possono anche trovarsi a grande distanza dai luoghi dove avviene la produzione agricola.

Oltre al dispositivo è necessario però poter disporre di un'interfaccia che consenta di leggere i dati ottenuti ed eventualmente elaborali. Questa viene spesso fornita da chi procura il dispositivo.

Può dunque verificarsi la condizione in cui, all'interno di una flotta, coesistano diverse tipologie di rilevatori, per cui colui il quale desideri monitorarla si troverà a dover impiegare a tal fine più software differenti.

Accade poi di dover ampliare la flotta o il numero di dispositivi presenti su ogni unità, oppure ancora di sostituirli: ciò vincolerebbe l'operatore ad una periodica sostituzione del software di gestione, o causerebbe un incremento del numero di applicativi da gestire.

L'obiettivo del presente studio è dunque lo sviluppo di un ambiente che sia in grado di memorizzare correttamente una mole di dati proveniente da un sistema di controllo flotte nella maniera più efficiente e razionale possibile. È necessario prevedere la ricezione ed il relativo

immagazzinamento di dati diversi (sia per quantità che per forma), e che il sistema sia pronto a ricevere qualsiasi tipo di dato senza dover essere preventivamente istruito sull'informazione che sta arrivando: è ad esempio possibile che si riceva la posizione GPS di un macchinario della flotta e subito dopo la quantità prodotta per agro di grano, oppure che vengano richieste le mappe con le caratteristiche di un terreno subito dopo.

È inoltre verificabile l'ipotesi in cui siano aggiunte nuove unità alla flotta con nuovi rilevatori: vi è perciò la necessità di poter implementare in maniera rapida l'ambiente di gestione, così da poter ricevere dati da nuove unità senza dover apportare modifiche strutturali.

Può -altresì- presentarsi l'eventualità che siano progettati nuovi dispositivi di rilevazione per parametri che non erano in precedenza controllati, o lo erano secondo una modalità differente: è dunque necessario prevedere che questi dispositivi vengano aggiunti correttamente alle unità della flotta.

Nasce così l'esigenza di poter aggiungere nel tempo nuove tipologie di rilevatori cui sia consentito l'invio di dati all'ambiente.

Anche la parte del sistema relativa alla definizione del tipo di dati recepibili deve - dunque - essere espandibile in maniera semplice, senza che siano necessarie modifiche complesse ed articolate, che rallenterebbero le operazioni ed appesantirebbero il sistema di gestione.

La finalità del progetto è pertanto definibile come la creazione di un ambiente di gestione dotato di eccezionale flessibilità, sia in relazione al tipo di dati che riceve ed immagazzina, sia alla quantità di fonti ammesse all'invio di dati.

Verrà dunque definito nelle seguenti sezioni dell'elaborato un ambiente di sviluppo per la creazione del sistema e saranno descritte le singole tecnologie che lo compongono.

Successivamente sarà determinata la struttura della base di dati deputata a contenere le informazioni che provengono dai rilevatori, e la struttura del software di gestione che dialogherà con esso, ipotizzando un prototipo di applicativo.

Seguirà una parte che descrive la realizzazione della proof of concept di tutto il sistema ideato, consistente un una terminale di rilevamento e trasmissione dati GPS dialogante con un sistema server remoto tramite rete TCP/IP dotato di piattaforma DBMS per gestione dati.

Riassumendo brevemente gli obiettivi che si vogliono raggiungere nella ideazione della architettura sono:

1. Ottimizzazione della gestione dei terminali installati sulla flotta
2. Dinamicità del sistema di controllo e gestione
3. Agevolazione delle modalità di ampliamento della flotta o dei sistemi di controllo

4. Nessun vincolo rigido sul numero di unità o di terminali
5. Nessun vincolo rigido sulla tipologia di terminali gestibili.

4.4 Schema generale Mobile Information System

Uno schema generale della architettura ideata - Schema Mobile Information System (MI System) - è stato inserito in Appendice A per migliorarne la leggibilità. In questo paragrafo lo schema viene brevemente discusso. Al centro dello schema è presente l'infrastruttura pensata per il trasporto dei dati, che non può essere che Internet considerando la necessità di raggiungere località geograficamente distanti, e la pervasività ed economicità consentita da questo mezzo. Per assicurare la necessaria riservatezza dei dati in transito sulla Rete pubblica nel Task 1 verrà analizzata e discussa una proposta che consentirebbe di risolvere il problema. Di seguito una breve descrizione delle componenti del sistema.

4.4.1 Mobile Information Base

Stazione base con connettività Internet ad alta disponibilità. La presenza di un server addizionale per interfaccia web è subordinata alla scelta di realizzare la parte di interfaccia grafica con accesso via http, piuttosto che tramite applicazione nativa del Sistema Operativo.
Infrastrutture realizzate con software Open Source. Il server MIS implementa comunicazione con terminali remoti eterogenei, e memorizza dati ricevuti in Relational Database Management System (RDBMS). Interfacce grafiche remote comunicano direttamente con RDBMS, con prestazioni che possono facilmente scalare ridondando le infrastrutture.
Nello schema in corrispondenza della MIB sono evidenziati i seguenti componenti:

1. Server memorizzazione dati con Sistema Operativo Open Source. In realtà per tale funzione è stato previsto di utilizzare un cluster di macchine collegate tramite Rete TCP/IP con terminali remoti. E' il cuore del sistema e deve quindi essere caratterizzato da alte prestazioni per essere in grado di gestire un elevato numero di accessi concorrenti sia per la memorizzazione che per la successiva elaborazione dei dati. Tipicamente sul cluster sarà installato un sistema RDBMS con un database ad architettura dinamica. Nel seguito verranno discussi più approfonditamente alcuni dettagli implementativi nella parte che descrive la proof of concept.
2. Server interfaccia web. Nel caso di un numero molto elevato di accessi contemporanei è possibile separare la funzione di RDBMS da quella applicativa nella gestione dei dati, e

quindi implementare un cluster di macchina deputate a far girare le applicazioni di gestione. Anche in questo caso Sistema Operativo e strumenti di sviluppo e produzione software possono essere completamente FLOSS. Ulteriori dettagli implementativi saranno descritti nella parte di proof of concept.

3. Workstation. Terminali locali per la gestione del MIS. Sebbene la parte di interfaccia sia stata immaginata da subito come remotabile, ciò non pregiudica la possibilità di utilizzare interfacce locali per massimizzare ad esempio la velocità di risposta dei terminali grafici. Anche qui Sistema Operativo e software a tutti i livelli FLOSS.

4.4.2 Terminali mobili

Terminali ad uso veicolare, con interfacce verso reti di tipo diverso, ad esempio automotive (CAN) e o agricole (ISOBUS), ed anche per la ricezione di segnali di vario genere, ad esempio GPS. Consentono tracking, funzioni di sicurezza ed antifurto, gestione di flotte e di dati di produzione. Ogni terminale comunica autonomamente con MIS server tramite connessione TCP/IP radio (GSM) o SMS. Il protocollo proposto nel Task 3 prevede anche funzionalità specifiche per comunicazioni long range per terminali mobili remoti.

4.4.3 Stazioni mobili

Il MIS è stato progettato con una architettura general purpose volta alla semplice integrabilità di terminali remoti con funzioni eterogenee. La sua architettura modulare consente di integrare rapidamente ed efficacemente nuove tipologie di terminali remoti. Per ottenere tale risultato è stato previsto di utilizzare esclusivamente sia per la parte hardware che software architetture aperte, ad esempio strumenti e applicativi Free, Libre and Open Source (FLOSS) per il software, ed inoltre la struttura dei servizi ed applicativa è stata pensata in modo che fosse il più possibile aperta, modulare e dinamica, in modo da rendere il più semplici possibili modifiche ed integrazioni.

4.4.4 Rete locale sensori

Rete locale di sensori con protocolli di diversa natura e stazione base che raccoglie dati e li trasmette al MIS server via connessione TCP/IP, radio (GSM) o SMS. Disposizione fisica dei nodi della rete non prestabilita, grazie a protocolli di comunicazione con routing adattivo. Le reti di sensori per quanto siano state nativamente considerate nel progetto sono fuori dallo

scope del presente documento, ed oggetto di lavoro da parte di altri gruppi ISO. In ogni caso la loro integrabilità non sarebbe critica utilizzando nella funzione di stazione base (gateway) un terminale conforme allo ‘standard’ MIS.

4.4.5 Interfacce grafiche remote

Interfacce grafiche con accesso remoto e specializzate per la funzione da svolgere, ad es. raccolta dati da sensori geografici oppure gestione flotte tramite terminali mobili. L’accesso via TCP/IP consente la raggiungibilità del server memorizzazione dati da qualsiasi posizione, unico requisito la connettività Internet. Opzionale implementazione interfaccia grafica via WEB.

4.5 Schema Mobile Information Server

Uno schema più dettagliato della parte server della architettura ideata - Schema Mobile Information Server (MI Server) - è stato inserito in Appendice A per migliorarne la leggibilità. In questo paragrafo lo schema viene brevemente discusso. Di seguito sono elencate alcune caratteristiche generali del sistema:

1. Software open source. Tutti i componenti del MIS sono basati su software FLOSS, così come il RDMBS ed i sistemi operativi su cui si basa il progetto. I componenti scelti sono caratterizzati da una ridondabilità nativa, in modo che sia possibile scalare semplicemente le prestazioni aumentando il numero di servizi concorrenti (ad es. server RDBMS o server WEB multipli).
2. Interfaccia Software open source web o applicazione. L’interfaccia operatore del MIS è implementata tramite applicazioni native ovvero opzionalmente via web. Possono essere adottate interfacce diverse specifiche per la funzione a cui sono destinate, ad es. semplice raccolta dati o tracking. Non costituisce nucleo del progetto di dottorato, verrà comunque realizzata una applicazione dimostrativa.
3. Relational DataBase Management System. Servizio di archiviazione dati basato su software standard ad elevate prestazioni. Nel MIS costituisce anche l’interfaccia verso l’esterno, poiché risulta efficiente non implementare un canale di comunicazione ad hoc. Le interfacce grafiche utente dunque comunicano direttamente con il RDBMS anche bidirezionalmente.

4.5.1 Mobile Information Server - caratteristiche e componenti

Poiché il MI Server è il nucleo del sistema ne viene qui fornita una descrizione più approfondita, partendo dalla sua architettura generale per poi discutere brevemente i suoi componenti fondamentali. Il MIS, scritto in C, si basa completamente su software Open Source. E' una applicazione multi-processo, in grado di gestire un elevato numero di connessioni contemporanee, e di scalare bene con hardware a core multipli. La struttura è fortemente modulare, in modo da rendere il sistema adattabile a scenari di vari natura. I principali componenti sono elencati nel seguito:

1. Master module. E' il modulo principale del sistema. In seguito ad ogni richiesta di comunicazione via TCP/IP (o SMS) ne viene generata una occorrenza, che si occupa di gestire una singola sessione di dialogo con terminale invocando a sua volta ulteriori moduli per soddisfare le richieste che gli giungono. Gestisce inoltre la comunicazione e gli errori.
2. Moduli di comunicazione - TCP/IP e SMS. Il primo si occupa di ricevere e smistare connessioni via TCP/IP o dal protocollo al Task 3 provenienti da terminali mobili. Una volta stabilita la connessione trasferisce la sua gestione al modulo gestore di protocollo. Il secondo si occupa di gestire comunicazioni via SMS provenienti da terminali mobili. Può essere utile nei casi in cui non sia presente o conveniente un connettività dati.
3. Moduli parser e config. Si occupano di fornire la configurazione o eseguire comandi remoti. Il caso tipico di comando è la richiesta di memorizzare nel RDBMS delle informazioni, ad esempio coordinate GPS per il tracking oppure dati tratti da una qualsiasi sorgente di informazioni ad Es . CAN o ISOBUS da terminali mobili o dati da rete locale sensori.
4. Modulo RDBMS. Si interfaccia con il RDBMS. Gestisce la selezione della struttura di database da utilizzare in funzione del dispositivo che richiede accesso. La comunicazione con il database viene in questo modo modularizzata permettendo di implementare un sistema general purpose, che consente la archiviazione e la gestione di dati provenienti da fonti eterogenee.

4.6 Proof of Concept - introduzione

A questo punto inizia la descrizione del sistema dimostrativo (Proof of Concept) progettato e realizzato. Sebbene il sistema non implementi tutte le caratteristiche del progetto iniziale, è da tener conto che insieme alle altre attività del presente dottorato è da ritenersi come facente parte di una proposta, avanzata diversamente per i tre Task per uno Standard Internazionale, e

quindi oggetto di confronto, dialogo e modifiche in un periodo di tempo che sicuramente travalica la durata di un normale dottorato. In ogni caso il sistema realizzato è funzionante ed implementa le funzionalità ed i concetti fondamentali del progetto originario, in modo da costituire una compiuta Proof of Concept, dimostrativa delle potenzialità del progetto compiuto nell'ambito del Task 0.

4.7 Proof of Concept - premessa

Nei seguenti paragrafi di questo capitolo si introdurranno alcuni concetti fondamentali sulle filosofie che hanno portato alla scelta degli strumenti utilizzati durante tutto il lavoro di ricerca oggetto del presente Dottorato. Tali argomenti sono qui discussi, e non saranno ripetuti, ma di fatto rappresentano le linee guida per molte scelte compiute nello svolgimento di tutti e tre i Task. Un tipico esempio è l'utilizzo di strumenti software afferenti al mondo dell'Open Source, ma anche l'utilizzo di architetture hardware preesistenti che sono standard aperti. E' evidente come tale tipo di approccio sia di fondamentale importanza nel momento in cui si desideri progettare una proposta che ambisce a diventare uno Standard Internazionale. Alcuni concetti fondamentali sono introdotti nel Task 0, visto che per omogeneità di trattazione non è parso utile inserirli come parte a sé stante, ma in realtà la filosofia di base ne permea l'intero progetto.

4.7.1 Proof of Concept - Il software FLOSS

Nella progettazione della Proof of Concept realizzata per il Task 0 si è scelto di avvalersi esclusivamente di software FLOSS. Tale dicitura indica un software i cui autori (più precisamente i detentori dei diritti) ne permettono, anzi ne promuovono, il libero studio e l'apporto di variazioni e migliorie tecniche da parte di altri programmatori; in realtà questo è sicuramente vero per la sottocategoria del software FLOSS detta 'Open Source'. La volontà di condivisione e collaborazione è applicata concretamente mediante l'impiego di apposite licenze d'uso. Per il software 'libero' in generale ciò può non essere sempre vero, nel senso che potrebbe darsi il caso che un software sia di utilizzo libero, ma non sia disponibile il codice sorgente. Nel caso dei software utilizzati nel presente Dottorato la scelta è sempre stata indirizzata verso casi in cui sia effettivamente disponibile e aperto il codice sorgente degli strumenti utilizzati, quindi tipicamente software 'Open Source'. La filosofia open source consente, grazie alla collaborazione di più soggetti (in genere di natura volontaria e spontanea,

anche se alcuni progetti Open Source sono in realtà espressione di grandi aziende), di conferire al prodotto finale un valore qualitativamente e tecnicamente più elevato di quanto potrebbe realizzare un singolo gruppo di lavoro o una singola azienda per quanto importante.
Inoltre, l'applicazione di licenze che ne consentano la libera diffusione e implementazione conferisce notorietà al prodotto e lo rende accessibile a tutti. Ciò ha permesso di utilizzare software altamente implementato e collaudato senza nessun tipo di costo, e ha consentito in fase di progettazione di poter prevedere anche qualche modifica al software qualora esso non rispondesse in maniera perfettamente aderente a talune delle esigenze progettuali. Particolarità del software Open Source oltre alla tipica totale disponibilità ed accessibilità del codice sorgente, che può così essere ottimizzato in funzione delle finalità del progetto in corso, è che nessun vincolo è posto al suo utilizzo da parte di chiunque, cosa di fondamentale importanza qualora l'obiettivo sia di costituire uno Standard Internazionale. Per una più approfondita discussione sulla terminologia e sulle varie tipologie di software che vengono comunemente indicate con il termine FLOSS, e sulle licenze ad esse collegate sono presenti alcuni riferimenti in bibliografia.

4.8 Proof of Concept - Definizione dell'ambiente di sviluppo e gestione

In questo paragrafo vengono delineate alcune delle scelte operate per l'ambiente di sviluppo utilizzato nella realizzazione della Proof of Concept.

4.8.1 Il Sistema Operativo

Partendo dal sistema operativo su cui appoggiare l'ambiente di sviluppo, esiste un grandissimo numero di varianti tra cui poter scegliere, per dare una idea della vastità delle scelte possibili basta osservare la seguente Figura 4.1:

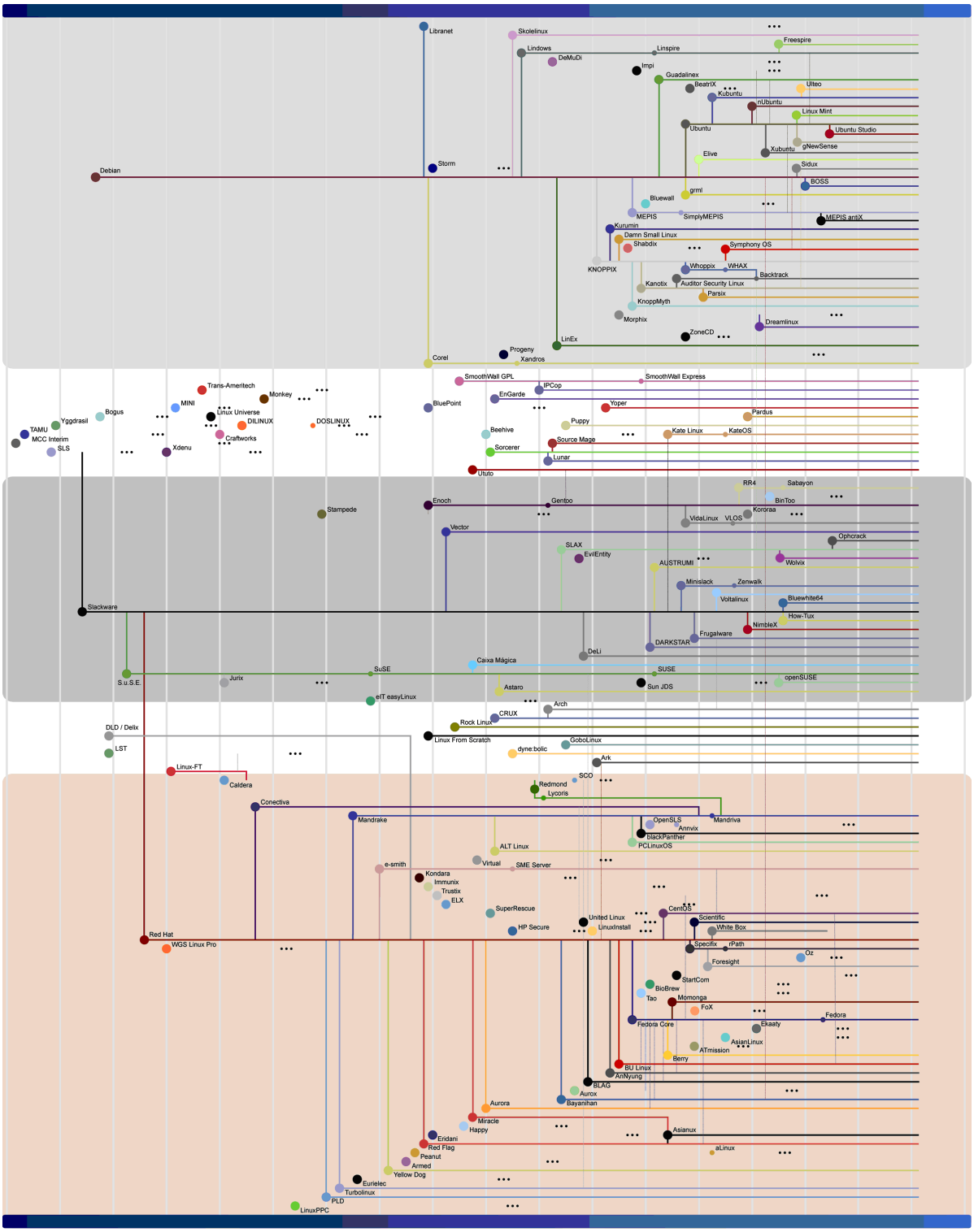


Figura 4.1: Albero delle più note distribuzioni GNU/Linux

Esistono in realtà molte altre versioni di Unix libere, ma la scelta è caduta sulla famiglia GNU/ Linux per le sue caratteristiche di grande diffusione, ampio utilizzo nelle realtà aziendali (dato che ne attesta la solidità e l'affidabilità) e il fatto che in questi ultimi anni tale sottocategoria sia diventato uno standard ‘de facto’ nel mondo dell’informatica. Inoltre questa famiglia di sistemi operativi ha anche il vantaggio di essere disponibile su di una ampia famiglia di dispositivi.

Altro elemento importante a favore di questo tipo di sistemi è data dalla possibilità di ottenere facilmente e gratuitamente supporto tramite la collaborazione con le numerose comunità on-line dedicate all'approfondimento pratico e all'uso di questo sistema operativo.

Anche restringendo il campo alla sola famiglia GNU/Linux vi sono molte alternative di ‘distribuzione’ potenzialmente adatte alla realizzazione di un ambiente di sviluppo, e per la implementazione dei componenti del sistema MIS. Per effettuare una scelta è dunque necessario valutare altri parametri.

Come si può notare dalla figura 4.1, le distribuzioni si suddividono in tre principali macro famiglie: Debian, Slackware e Red Hat. Tra questi tre filoni fondamentali è stata selezionata Red Hat in base alle seguenti considerazioni.

1. Red Hat ha affiancato i prodotti con un servizio di assistenza progettato per aziende anche di importanti dimensioni, il mercato cosiddetto Enterprise: questo ha avuto come effetto il fatto che molte realtà aziendali si sono orientate verso questo tipo di prodotto. Di conseguenza la comunità open source che si rivolge alle aziende ha implementato sempre di più questo sistema, e si è creato un indotto grazie al quale oggi è possibile trovare praticamente la totalità delle applicazioni rivolte alle esigenze produttive delle aziende compatibili con Red Hat. Molte di esse sono disponibili anche per altre piattaforme, ma Red Hat è indiscutibilmente tra tutte quella più diffusa. Considerando che le necessità aziendali sono comunemente orientate verso la stabilità e la sicurezza, si è valutato che le necessità del sistema da implementare fossero conformi a quelle aziendali.
2. La grande diffusione dell’ecosistema RedHat Enterprise ha avuto effetto anche sul mondo scientifico ed accademico, per cui si sono diffuse distribuzioni basate su tale tecnologia anche in questi ambiti. Un tipico esempio è Scientific Linux, che è una ‘versione’ di Linux messa insieme dal laboratorio Fermilab del CERN, e da vari altri laboratori ed Università in tutto il mondo. Il suo obiettivo primario è ridurre la duplicazione degli sforzi nella creazione di un S.O. scientifico affidabile, e di avere una base comune di installazione per i

vari esperimenti e relativi sperimentatori. La distribuzione Scientific Linux è di fatto una RedHat Enterprise ricompilata partendo dai sorgenti.

3. Red Hat Enterprise Linux è in grado di soddisfare i requisiti tecnici e funzionali presenti nel settore delle telecomunicazioni, elemento essenziale per il progetto, in quanto la maggioranza delle fonti da cui perverranno i dati da immagazzinare saranno geograficamente distanti dal sistema; è stato dunque necessario prevedere l'utilizzo di tecnologie tipiche del settore delle telecomunicazioni.
4. Alcune delle tecnologie di punta di RedHat sono state utilizzate anche nei Task successivi, ad esempio il supporto al nuovo Kernel Real-Time di cui si tratterà più ampiamente nella parte riguardante il Task 2.
5. Red Hat Enterprise Linux è una delle distribuzioni GNU/Linux più stabili attualmente presenti sul mercato, ed ha generato una serie di derivate liberamente utilizzabili ed adatte a vari scopi.

Quest'ultimo punto è molto importante, nell'ottica di costruire uno Standard Internazionale. Adottare un modello Open Source consente anche di mantenere la possibilità di scegliere tra varie implementazioni di uno strumento, a seconda delle necessità del singolo progetto in termini di supporto, tecnologie da utilizzare e adattabilità ad un particolare scopo. Ad esempio se le esigenze progettuali non richiedono i servizi di assistenza e di consulenza forniti da Red Hat, è possibile optare per una delle distribuzioni completamente gratuite derivate. Nel caso invece si ritenesse utile è sempre possibile scegliere diversamente, senza però dover riprogettare tutte le applicazioni, che continueranno con tutta probabilità a funzionare anche su piattaforme differenti ma compatibili. Ciò ovviamente è un enorme vantaggio in quanto consente una totale libertà attuale e futura nella scelta del fornitore.

Tra le varie alternative disponibili su base RedHat nel caso specifico delle presente Proof of Concept è stata selezionata CentOs. Questa è una distribuzione che è stata sviluppata con la finalità di rendere disponibile una piattaforma enterprise libera a chiunque voglia usarla senza dover necessariamente sostenere i costi di una licenza Red Hat.
Le sue release sono compilate a partire dai sorgenti SRPMS open source pubblici di un 'prominent North American Enterprise Linux vendor', cioè Red Hat.

CentOS è completamente compatibile con il prodotto Red Hat, e tendenzialmente modifica il sorgente originale unicamente al fine di rimuovere e/o modificare le customizzazioni grafiche e gli artwork di Red Hat. In ogni caso né il CentOs Project né qualsiasi versione di CentOs sono in alcun modo collegate con l'azienda Red Hat.
In base al complesso di queste osservazioni si è scelto di appoggiare l'ambiente di sviluppo e per semplicità anche quello applicativo su piattaforma CentOs.

4.8.2 Eclipse - Ambiente di Sviluppo Integrato

Eclipse è un IDE (Integrated Development Environment), consiste cioè in una piattaforma integrata che consente di gestire l'intero processo di sviluppo di applicazioni di livello enterprise. L'architettura di base, dal valore iniziale di circa 40 milioni di dollari, è stata donata da IBM alla comunità open source attraverso la costituzione di un consorzio che vede in prima linea, oltre a IBM, aziende leader del settore ITC: Borland, Merant, Qnx Software, Rational Software, Red Hat, Suse, Togethersoft e Webgain. La filosofia progettuale di Eclipse è tesa all'obiettivo di dotare gli sviluppatori di strumenti di sviluppo sempre più sofisticati e potenti, grazie ad un framework unico che integri potenzialmente tutti i linguaggi, gli Ide, i tool di modeling e gli strumenti di collaborazione esistenti. Si tratta di un framework basato su un sistema di API pubbliche e di plugin le cui specifiche di sviluppo sono di pubblico dominio. Ciò consente a chiunque di sviluppare i propri plugin e renderli disponibili alla comunità open source, oppure di crearne versioni particolari a pagamento. Eclipse viene abitualmente impiegato da moltissimi sviluppatori in tutto il mondo ed è dotato, già dalle precedenti versioni, di caratteristiche di qualità e produttività per la realizzazione di applicazioni enterprise fino a poco tempo fa inimmaginabili, anche su prodotti di mercato dal prezzo elevato.
Per questo progetto il suo utilizzo è stato di particolare rilevanza e pregio, ed ha consentito un gran risparmio di tempo nella stesura del codice, grazie alla sua funzione integrata di debug ed alla sua gestione complessiva dell'ambiente di programmazione.
Ancora di più sono evidenti i vantaggi nella adozione di uno strumento del genere per la realizzazione di uno Standard Internazionale, in quanto non solo è già uno standard di fatto nel mondo delle Information e Communication Technologies (ICT), ma in più è gratuitamente disponibile, liberamente modificabile e personalizzabile da chiunque.

4.8.3 Il Relational Database Management System

Un Relational Database Management System (RDBMS) consiste in un sistema software basato sul modello relazionale, progettato al fine di consentire la creazione e manipolazione efficiente di database (ovvero di collezioni di dati strutturati) tipicamente da parte di più utenti. Poiché il progetto si occupa delle operazioni di ricezione e immagazzinamento di dati, finalizzato ad una successiva fruizione degli stessi per molteplici finalità operative, l'RDBMS sul quale si appoggia è una delle parti fondamentali, anzi è insieme alla architettura del database vero e proprio, da ritenersi il cuore del sistema.
È stato perciò necessario selezionare con particolare cura il prodotto, conformemente alle suddette esigenze progettuali. Nel seguito sono forniti alcuni dettagli inerenti il processo di scelta.

4.8.3.1 Scelta del RDBMS

Tenendo conto delle considerazioni sinora effettuate riguardo la scelta di soluzioni Open Source, molti tra i prodotti più diffusi e maturi sono stati considerati.

	MySQL 5	PostgreSQL	SQLite
Max DB size	Unlimited	Unlimited	32 TB (230 pages * 32 KB max page size)
Max table size	2 GB (Win32 FAT32) to 16 TB (Solaris)	32 TB	?
Max row size	64 KB	1.6 TB	?
Max columns per row	3398	250-1600 depending on type	2000
Max Blob/Clob size	4 GB (longtext, longblob)	1 GB (text, bytea) - stored inline	1 GB
Max CHAR size	64 KB (text)	1 GB	1 GB
Max NUMBER size	64 bits	Unlimited	64 bits
Min DATE value	1000	-4713	No DATE type
Max DATE value	9999	5874897	No DATE type

Tabella 4.2: Limiti di alcuni noti RDBMS Open Source

Di seguito sono discussi a titolo di esempio i termini che hanno portato alla scelta finale, tra tre dei concorrenti ritenuti più interessanti, ovvero MySQL, PostgreSQL e SQLite. Osserviamo in primo luogo i limiti tecnici (relativi al momento in cui si è operato il confronto) dei tre database per compararli in Tabella 4.2. Da tale tabella si può notare che MySQL e PostgreSQL non hanno limiti alla dimensione massima del database mentre SQLite ha un limite.
Tale limite, anche se può sembrare particolarmente elevato, rappresenta per il progetto una rilevante limitazione, poiché il database sarà continuamente alimentato dai dati inviati da un numero arbitrario di terminali remoti e dispositivi di vario genere: per questo una limitazione alla dimensione del database potrebbe costringere all'impiego di più di un database e alla conseguente revisione ed adeguamento di tutte le strutture di controllo, ricerca e inserimento dei dati. Tale situazione non è ovviamente desiderabile nel caso si voglia realizzare uno Standard Internazionale e quindi si pensi a sistemi con meno vincoli possibili. E' anche vero che in astratto è possibile immaginare una architettura indipendente dai dettagli implementativi del singolo RDBMS, ma tale situazione è ideale, nella realtà conviene tener conto in fase di progetto di eventuali limitazioni imposte. Tale limitazione della quota del database in SQLite è dunque un elemento la cui rilevanza ai fini progettuali non è da non sottovalutarsi. Per quanto riguarda le altre tipologie di limitazioni dei database possono considerarsi meno vincolanti per la scelta dell'uno o dell'altro prodotto, ma in ogni caso è da considerare la generale migliore performance di PostgreSQL.
Vediamo ora in Tabella 4.3 i principali elementi che caratterizzano un RDBMS:

	MySQL	PostgreSQL	SQLite
ACID	Yes	Yes	Yes
Referential integrity	Yes	Yes	No
Transactions	Yes	Yes	Basic
Unicode	Partial	Yes	Yes
Interface	SQL	SQL	SQL

Tabella 4.3: Caratteristiche degli RDBMS

Come si può notare tutti e 3 i RDBMS hanno le proprietà ACID (Atomicità, Coerenza, Isolamento, Durabilità), essenziali per un sistema qualitativamente performante.
SQLite però non ha pieno supporto all'integrità referenziale, e questo rappresenta un elemento negativo. Dato che, in fase operativa, saranno diversi i terminali e gli utenti che accederanno contemporaneamente al database, l'integrità referenziale diventa una caratteristica assai determinante, in quanto garantisce che, nel caso di numerosi accessi contemporanei alla stessa tabella, non si verifichino conflitti (ad esempio impedisce la generazione di record duplicati in un campo chiave).
Nella fattispecie, una sorta di integrità referenziale può essere assicurata anche grazie all'impiego dei triggers, ma si tratta comunque di una modalità di gestione più macchinosa, e che richiede comunque un costante monitoraggio.
In relazione alle altre caratteristiche, non sono osservabili differenze di entità tale da poter incidere in maniera determinante sulla scelta del prodotto, anche se permane in generale una performance migliore di PostgreSQL.
Osserviamo ora le principali funzionalità aggiuntive che possono essere previste in un RDBMS in Tabella 4.4:

	MySQL	PostgreSQL	SQLite
Union	Yes	Yes	Yes
Intersect	No	Yes	Yes
Except	No	Yes	Yes
Inner joins	Yes	Yes	Yes
Outer joins	Yes	Yes	Yes
Inner selects	Yes	Yes	Yes
Merge joins	Yes	Yes	?
Blobs and Clobs	Yes	Yes	Yes

Tabella 4.4: Principali funzionalità aggiuntive degli RDBMS

Da questa tabella possiamo notare che MySQL non gestisce l' "Intersect" e l' "Except".

Except è un operando che è possibile inserire tra query in una riga di comando. Come risultato si ottengono tutti i valori distinti della query a sinistra dell'operando non presenti nella query a destra.
Intersect è anch'esso un operando che è possibile inserire tra query in una riga di comando. Il risultato è però formato da tutti i valori distinti restituiti da entrambe le query a sinistra e a destra dell'operando.
Poiché non è ai fini progettuali sostanzialmente rilevante stabilire come i dati immagazzinati nel sistema saranno successivamente fruiti, non è possibile valutare a priori se tale elemento potrà costituire una effettiva limitazione all'ambiente.
È doveroso osservare che i risultati ottenibili da questi due operandi possano altresì essere ottenuti in altro modo, tramite cioè i comandi disponibili su MySQL, ma tale procedimento appare più macchinoso e complesso: possiamo perciò considerarlo un elemento a sfavore di MySQL.
Si può dunque rilevare che, in relazione alle funzionalità aggiuntive, non si apprezzano sostanziali differenze in riferimento al progetto, se non una lieve scomodità nell'applicazione pratica di MySQL.
Dalla precedente analisi possiamo soltanto dedurre che l'RDBMS SQLite pare non essere il prodotto più adatto al nostro tipo di progetto.
Ma per operare una scelta tra MySQL e PostgreSQL occorre ottenere un altro parametro: la velocità del sistema. Vi è una particolarità che differenzia in maniera sostanziale MySQL da PostgreSQL: l'uso del "lock".
Quando un utente interagisce con i dati, il sistema li blocca eseguendo un lock, in modo che questi non siano accessibili da altri utenti. Tale protezione è indispensabile per evitare il verificarsi di accessi contemporanei che modifichino lo stesso dato, e venga così minata la coerenza del database.
La principale differenza sta nel fatto che MySQL esegue il lock a livello di tabella (mentre un utente sta interagendo con una tabella, questa viene interamente inibita all'utilizzo da parte di altri utenti), PostgreSQL invece lo esegue a livello di record (mentre un utente interagisce con una riga di una tabella, gli altri utenti non possono contemporaneamente interagire con la medesima riga).
Il lock a livello di record è assai più lento rispetto quello a livello di tabella, poiché è necessario prima scegliere il record da bloccare (utilizzando il comando select) e poi

bloccarlo. Tale operazione avviene in maniera completamente trasparente per l'utente ma causa altresì un rallentamento del sistema.
Di contro, con il lock a livello di record, se due utenti cercano di modificare dati nella stessa tabella ma in record differenti non è necessario che attendano che la tabella venga rilasciata dal lock.
Tale meccanismo crea una differenza importante: MySQL è più veloce se le dimensioni sono piccole e vi è un singolo utente, man mano però che le dimensioni e gli utenti aumentano, PostgreSQL diventa sempre più performante mentre MySQL risente sempre dei medesimi tempi di attesa.
Poiché nel progetto è prevista la presenza di molti terminali all'interno di una flotta e un numero di utenti potenzialmente altrettanto elevato, il fatto che un sistema scali bene verso l'alto, sia cioè sempre più performante con l'aumentare degli utenti, è un parametro di profonda rilevanza, che ha dunque fatto pendere pesantemente l'ago della bilancia nella scelta di PostgreSQL.
I ragionamenti sin qui esposti sono esemplificativi del complesso processo necessario nella valutazione di ogni singolo componente quando si intenda progettare uno standard. Alcuni dettagli implementativi per quanto possano sembrare poco inerenti il discorso generale della standardizzazione ne possono minare la effettiva applicabilità rendendo vano il lavoro di standardizzazione. Per questo motivo è necessario procedere con la massima attenzione considerando il maggior numero possibile di dettagli teorici e di implementazione, al fine di produrre uno Standard realmente applicabile.

4.8.3.2 Caratteristiche di PostgreSQL

In bibliografia sono presenti alcuni riferimenti che consentono di approfondire la conoscenza di PostgreSQL, sono qui fornite alcune indicazioni sulle sue caratteristiche più utili, e che forniscono la misura di quale contributo possa fornire la scelta di software Open Source allo sviluppo di qualsiasi progetto, ed in particolare nel caso dello sviluppo di uno standard.
PostgreSQL è un potente RDBMS open source che, con i suoi più di quindici anni di sviluppo attivo e architettura testata, si è guadagnato una eccellente reputazione in relazione ai parametri di affidabilità, correttezza ed integrità dei dati. È disponibile per la maggior parte dei sistemi operativi ed ha uno spettro dei data type molto ampio, ciò significa che è consentito

immagazzinare i dati all'interno di tabelle con una forma molto più adattabile ai tipi di dati esistenti.
In molti casi infatti si ricorre a campi testuali per contenere dati che invece potrebbero essere classificati con maggior precisione, in modo tale da garantirne una gestione più corretta (ad esempio se un campo testuale viene usato per contenere delle date, poi non si potrà pensare di ordinare quei dati per mese o per giorno, ma potranno solo essere ordinati in base all'intera stringa che contiene la data).
Inoltre PostgreSQL contiene un framework che consente agli sviluppatori di creare tipi di dati personalizzati a seconda delle esigenze. Ciò è molto rilevante se si immagina di costruire un database che sia molto dinamico e senza ipotesi iniziali sulla tipologia di dati da inerire. Tale caratteristica dunque è molto interessante in quanto consente una ottima flessibilità.
Una caratteristica interessante ancora è il supporto per l'archiviazione di oggetti di grandi dimensioni binarie, incluse immagini, suoni e video.
Tale sistema può vantare un ottimo supporto dello standard SQL ANSI-SQL 92/99, nonché delle subqueries (che permettono di strutturare le query evitando ridondanze e inutili perdite di tempo), e ha un ottimo sistema di indici chiamato GiST (Generalized Search Tree). GiST prevede infatti la specifica del tipo di dato, la modalità della sua archiviazione, e la possibilità di definire parametri di ricerca su un determinato tipo di dato.
Una interessante particolarità di PostgreSQL consta nel carattere di ereditarietà delle tabelle, è infatti possibile, sulla base di tabelle già esistenti, crearne di nuove che ne ereditino la medesima struttura, con la possibilità di ampliarla. Per fare questo è sufficiente, in fase di creazione della tabella, fare riferimento alla tabella che ha la struttura di base che si desidera ereditare.
Per quanto riguarda la personalizzazione, PostgreSQL può vantare una grande elasticità. Oltre all'estensibilità dello spettro dei tipi di dato è possibile utilizzare delle stored procedure (procedure per estendere le potenzialità del sistema) che possono essere scritte in dozzine di linguaggi di programmazione differenti (incluso Java, Perl, Pyton, Ruby, TCL, C/C++, e il proprietario PL/pgSQL). È previsto inoltre l'utilizzo di triggers che, uniti alle procedure, possono essere richiamati come librerie scritte in C all'interno del database.
Queste caratteristiche fanno di PostgreSQL uno strumento di assoluto spessore tecnico, che conferisce solidità e allo stesso tempo flessibilità a tutto l'ambiente di gestione.

4.9 Proof of Concept - Il Database

Data la caratteristica di estrema flessibilità dell'intero sistema di gestione, la struttura del database dovrà necessariamente essere complessa. Nella Proof of Concept è implementata una struttura che implementa la principali caratteristiche richieste ad un sistema come il MIS, in modo tale da presentare un sensato punto di partenza per sviluppi successivi.
La prima esigenza a cui deve rispondere il database è di prevedere la possibilità di ricevere ed archiviare i dati inviati da più unità della flotta, ed anche da più terminali installati sulla stessa unità. Si pone quindi la questione di quali terminali e quali unità possano essere abilitati.
La risposta a questo interrogativo deve essere necessariamente rappresentata all'interno del database.
In questo ambiente di gestione si è scelto di abilitare all'invio di informazioni tutti i terminali che siano mai stati inizializzati (per essere inizializzati i terminali devono comparire in un elenco di "tipi" consentiti per l'inizializzazione). Questo perché il sistema deve essere in grado di riconoscere il terminale che invia i dati, e deve sapere in quale modo archiviare le informazioni ricevute.
Inoltre, poiché queste informazioni saranno sicuramente eterogenee sia nella quantità che nella forma, sarà indispensabile prevedere più strutture in grado di riceverle.
Si potrà quindi dividere il database in due parti fondamentali:

1. gestione periferiche, che comprende le tabelle e le relazioni necessarie a gestire i terminali, la loro inizializzazione e la loro identificazione;
2. gestione dati, che comprende le tabelle e le relazioni necessarie a ricevere e immagazzinare i dati ricevuti dalla flotta.

Appare quindi evidente che la struttura della parte gestione periferiche potrà essere fissa, mentre la parte di gestione dati avrà necessariamente carattere dinamico, perché non sarà possibile stabilire a priori nemmeno il tipo di dato che verrà inviato all'ambiente di gestione.
Entriamo nel merito utilizzando il modello E/R (Entity/Relationship) per la descrizione dei database, nel cui contesto verranno analizzati i seguenti elementi.

1. Entità. Dette anche tabelle, rappresentano classi di oggetti (fatti, cose, persone,...) aventi proprietà comuni ed esistenza autonoma ai fini dell'applicazione di interesse. Un'occorrenza di una entità consiste in un oggetto della classe che l'entità stessa rappresenta. Non si

intende -in questo caso- il valore che identifica l'oggetto, bensì l'oggetto stesso. Una interessante conseguenza di tale metodo è che un'occorrenza di entità ha una esistenza indipendente dalle proprietà ad esso associate. In uno schema ogni entità ha un nome che la identifica in modo univoco, ed è rappresentata graficamente tramite un rettangolo.

2. Associazioni. Dette anche relazioni, esse rappresentano un legame tra due o più entità.
   Il numero di entità legate è chiamato grado dell'associazione: un buon schema E/R è caratterizzato da una prevalenza di associazioni con grado 2.
   È possibile sia legare una entità a se stessa (attraverso un'associazione ad anello), sia legare le stesse entità con più associazioni.
3. Attributi. Si usano per descrivere le entità. Tutti gli oggetti della medesima classe entità hanno gli stessi attributi: questo è ciò che si intende per oggetti simili.
   La scelta degli attributi riflette il livello di dettaglio con cui vogliamo rappresentare le informazioni sulle entità. Per ogni attributo associato ad una classe entità è necessario definire un dominio di possibili valori. Per l'attributo nome, ad esempio, il dominio potrebbe consistere nell'insieme delle stringhe di 15 caratteri.
   Per ciascuna classe entità è possibile definire anche una chiave, un insieme minimale di attributi che identificano in maniera univoca una tupla (riga) all'interno del database. È consentita la presenza di più di una chiave: in questo caso si parla di chiavi candidate.
   La scelta deve altresì ricadere su una sola chiave candidata, detta chiave primaria. L'attributo si identifica -graficamente- con una ellisse al cui interno è specificato il nome dell'attributo, o anche (nel caso di diagrammi complessi) con la sola indicazione del relativo nome.
4. Cardinalità delle associazioni. Sono specificate per ciascuna delle entità partecipanti ad una associazione, ed indicano quante volte, in una relazione tra entità, un'occorrenza di una di queste possa essere legata ad occorrenze delle altre entità coinvolte nell'associazione (esprime cioè il minimo e il massimo delle occorrenze).

4.9.1 Database - Gestione periferiche

Questa parte del database deve contenere tutto il necessario alla gestione dei terminali.
È necessario quindi che consenta di:
- definire un elenco di tipi di terminali ammessi all'invio di informazioni
- controllare se un determinato tipo sia o meno presente nella lista di quelli consentiti

- inserire dinamicamente nuovi tipi, per consentirne la gestione
- definire in quale modo vengono trasmesse informazioni dai vari tipi di terminali

È altrettanto necessario poter:
- definire un elenco di periferiche inizializzate e quindi abilitate all'invio
- controllare se una determinata periferica sia già stata inizializzata o meno
- inizializzare una periferica.

Queste necessità si traducono in due entità: Tipi e Periferiche_inizializzate.

L'entità Tipi deve contenere un attributo ID_Tipo di tipo contatore numerico, che si autoincrementa ogni qual volta venga aggiunto un nuovo tipo: questo campo è la chiave primaria dell'entità. Ciò significa che tramite il numero progressivo contenuto in questo campo è possibile riferirsi univocamente ad un tipo.
Tipo_periferica, invece, è l'attributo contenente la stringa di testo che identifica univocamente il tipo di periferica. È qui che viene inserito l'insieme di caratteri con il quale il rilevatore identifica il proprio tipo.
È particolarmente importante che Tipo_periferica non ammetta duplicati, in quanto, se ci fosse più di una riga che identifica lo stesso tipo, si genererebbe confusione e si rischierebbe di inizializzare più volte lo stesso terminale, non sapendo a quale terminale attribuire correttamente i dati che arrivano.
Ad esempio, se il tipo "GPS" avesse come ID_Tipo sia 1 che 47, alla richiesta di inizializzazione di una nuova periferica di tipo GPS non si saprebbe se inizializzarla con identificativo 1 o 47: si rischierebbe di averla inizializzata con entrambi, e si avrebbe confusione perché ad ogni ricezione di dati il sistema non saprebbe a quale dei due identificativi farli corrispondere.
Questa necessità di non avere duplicati, se specificata nell'ambiente di gestione tramite un controllo precedente all'inserimento di un nuovo tipo, non sarebbe stata comunque sufficientemente rispettata in quanto, considerando la possibilità di richieste concomitanti di aggiunta dello stesso tipo, il sistema avrebbe potuto segnalare più volte che il tipo non è presente e avrebbe potuto accogliere contemporaneamente più richieste di inserimento, generando dei duplicati.

Esiste invece in PostgreSQL una proprietà chiamata "UNIQUE", assegnabile ad un attributo, che consente che sia il RDBMS stesso a verificare l'unicità dei valori.
Tipo_periferica è quindi "UNIQUE".
L'ultimo attributo di questa relazione è Istruzioni_inizializzazione.
In esso viene salvato l'elenco degli attributi necessari ad una relazione per contenere i dati inviati da un terminale del tipo in questione.
Ad esempio, per il tipo "GPS", Istruzioni_inizializzazione potrà avere come valore "a int, b int, c int, d int" che sta a specificare quattro attributi che possono contenere dei numeri interi (sufficienti a specificare una posizione GPS tridimensionale).
È molto importante che i valori di questo attributo vengano scritti con la seguente sintassi: nome_attributo1 spazio tipo_attributo1 virgola spazio nome_attributo2 etc, fino all'ennesimo ed ultimo attributo senza virgola e spazio finali.
Questo poiché il contenuto dell'attributo Istruzioni_inizializzazione viene utilizzato come parte di un comando SQL per creare un'entità del database, e, se il valore non viene inserito correttamente, la tabella non potrà essere creata, o non sarà adatta a contenere le informazioni che verranno inviate.
La relazione Periferiche_inizializzate deve contenere un attributo ID_Periferica di tipo contatore numerico, che si autoincrementi ad ogni inizializzazione di una nuova periferica. Questo campo è la chiave primaria dell'entità: ciò significa che, tramite il numero progressivo contenuto in questo campo, è possibile riferirsi univocamente ad uno dei terminali presenti su una delle unità della flotta.
È necessario inoltre che contenga un attributo ID_Tipo_riferimento che riporti lo stesso numero utilizzato nella tabella Tipi per indicare il tipo della periferica in oggetto.
Ad esempio, se è stato inserito il Tipo "GPS" e gli è stato assegnato in ID_Tipo il numero progressivo 25, tutte le periferiche di tipo "GPS" che verranno inizializzate riporteranno 25 in ID_Tipo_riferimento.
Possiamo quindi affermare che questo attributo rappresenta la chiave esterna collegata all'entità Tipi.
L'ultimo attributo della tabella è Periferica_seriale. In esso viene memorizzata una stringa di caratteri che riporta il numero seriale della periferica. Per evitare duplicati è necessario controllare che non esistano due periferiche inizializzate aventi lo stesso ID_Tipo_riferimento e anche il medesimo Periferica_seriale.

È infatti impossibile che esistano due periferiche del medesimo tipo e con uguale seriale (teoricamente è assai improbabile che esistano due periferiche con lo stesso seriale anche senza coinvolgere il tipo, però, considerando le ampie possibilità di espansione del database e il numero e l'eterogeneità di terminali con i quali si potrà avere a che fare, questa eventualità è stata in ogni caso prevista e valutata).
La coppia ID_Tipo_riferimento e Periferica_seriale è quindi "UNIQUE".
Quando si assegna questa proprietà ad una coppia di attributi, ciò significa che è possibile per ognuno dei due avere dei duplicati, ma non è possibile che esistano due tuple della tabella contenenti lo stesso valore per un attributo e che abbiano, nel contempo, un altro valore comune per l'altro attributo.
La relazione che lega queste due entità è, come esplicato, tra l'attributo ID_Tipo di Tipi e l'attributo ID_Tipo_riferimento di Periferiche_inizializzate, ed è di tipo uno a molti. La cardinalità dalla parte di Tipi è uno, mentre la cardinalità dalla parte di Periferiche_inizializzate è n. Ciò significa che ad ogni tipo di periferica possono corrispondere n periferiche inizializzate.
Di seguito nella Figura 4.5 è possibile vedere una rappresentazione grafica di questa parte del database.

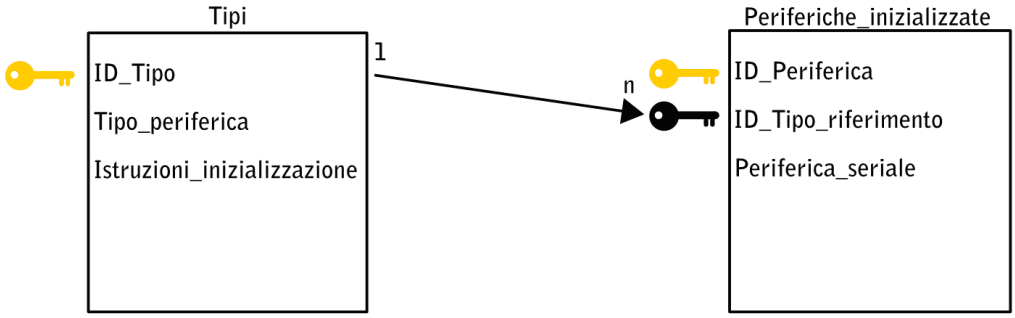


Figura 4.5: Database - Relazioni tabelle Tipi e Periferiche_inizializzate

A titolo di esempio il codice utilizzato per la creazione delle due tabelle è il seguente:

CREATE TABLE Tipi (ID_Tipo SERIAL PRIMARY KEY,
Tipo_periferica varchar(20) UNIQUE, Istruzioni_inizializzazione varchar);

CREATE TABLE Periferiche_inizializzate (ID_Periferica SERIAL PRIMARY KEY,
ID_Tipo_riferimento int REFERENCES Tipi (ID_Tipo), Periferica_seriale varchar(40),
UNIQUE (ID_Tipo_riferimento, Periferica_seriale));

4.9.2 Database - Gestione Dati

Questa parte del database deve contenere tutto il necessario alla gestione delle informazioni ricevute dai terminali.

Deve dunque consentire di:
- identificare quale sia la periferica che ha inviato i dati
- avere una struttura dati adeguata ad ogni tipo di periferica abilitata all'invio dati
- immagazzinare correttamente i dati ricevuti

Tali condizioni tecniche non possono essere soddisfatte da un numero predefinito di tabelle ed è questa, appunto, la parte più dinamica del database.
Per il controllo sulla periferica che ha inviato i dati, infatti, si fa riferimento alla tabella Periferiche_inizializzate descritta nel paragrafo precedente.
Per quanto riguarda i dati si rende altresì necessaria una struttura adattabile e variabile nel tempo, a seconda delle esigenze.
Ogni volta che verrà aggiunto un nuovo tipo di periferica, sarà dunque necessario creare una struttura dati adatta a contenere i dati che possono essere inviati da quel determinato tipo di periferica.
Tutte le entità create a questo scopo devono avere un nome uguale al tipo di periferica di cui contengono i dati e contenere un attributo ID_Periferica_riferimento che riporti il medesimo numero che utilizzato nella tabella Periferiche_inizializzate per identificare la periferica in questione.
Ad esempio, se è stato inserita la periferica "ABD79L" di Tipo "GPS" e gli è stato assegnato in ID_Periferica il numero progressivo 50, tutti i dati inviati da quella periferica si troveranno nell'entità "GPS" e riporteranno 50 come valore dell'attributo ID_Periferica_riferimento. Questo attributo è dunque definibile come la chiave esterna collegata all'entità Periferiche_inizializzate.
Inoltre, queste entità devono contenere tutti gli attributi elencati nel campo Istruzioni_inizializzazione, nella tabella Tipi.
Se, riportando l'esempio indicato nel paragrafo precedente, si volesse inserire il tipo "GPS", Istruzioni_inizializzazione potrà avere come valore "a int, b int, c int, d int", che sta a

specificare quattro attributi che possono contenere numeri interi (sufficienti a specificare una posizione GPS tridimensionale)
Conseguentemente si avrebbe una entità denominata "GPS" con cinque attributi: ID_Periferica_riferimento, a, b, c e d.
La relazione che lega questa entità all'altra parte del database, è tra l'attributo ID_Periferica_riferimento e l'attributo ID_Periferica di Periferiche_inizializzate ed è di tipo uno a molti. La cardinalità dalla parte di Periferiche_inizializzate è uno, mentre la cardinalità dalla parte di GPS è n. Ciò significa che ad ogni periferica di tipo GPS possono corrispondere n tuple di dati inviati.
Di seguito in Figura 4.6 è possibile vedere una rappresentazione grafica di questa parte del database:

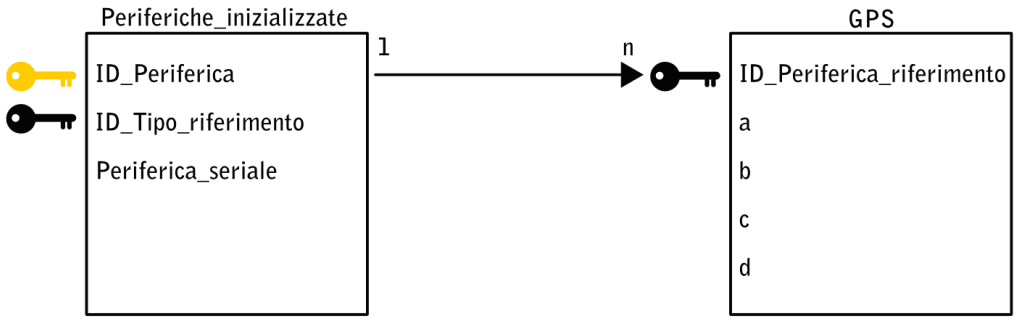


Figura 4.6: Database - Relazioni tabelle Periferiche_inizializzate e GPS

La sintassi utilizzata per creare questa entità sarebbe:

CREATE TABLE GPS (
ID_Periferica_riferimento int REFERENCES Periferiche_inizializzate(ID_Periferica),
a int, b int, c int, d int);

In Appendice B è presente una rappresentazione grafica di tutto il database compresa una tabella generica che indica la collocazione di ogni nuovo tipo aggiunto alla lista dei tipi consentiti per l'inizializzazione.

4.10 Proof of Concept - Software di gestione del Database

Parallelamente al database, un elemento fondamentale del progetto è rappresentato dal software di gestione. Questo software deve infatti creare una interfaccia tra il database e la flotta e deve consentire la gestione di tutte le richieste pervenute dai rilevatori.

Le richieste possibili sono principalmente tre:
- Inserimento di un nuovo tipo nell'elenco dei tipi di periferiche che possono essere inizializzate
- Inizializzazione di una nuova periferica
- Inserimento di dati

È inoltre necessario che sia prevista una corretta e dettagliata gestione degli errori, poiché, qualora un errore impedisse l'inizializzazione di una periferica, oppure l'inserimento di una serie di dati, è importante che sia dato di conoscere la natura e tipologia dell'errore, quando l'errore si è verificato, ed infine quale è stata la richiesta che lo ha generato.

4.10.1 Il linguaggio di programmazione

Come precedentemente accennato, l'utilizzo di PostgreSQL ha permesso di poter scegliere il linguaggio di programmazione con il quale implementare il software di gestione tra tutti quelli più diffusi e utilizzati.
Dato che l'applicativo da sviluppare era di sistema, e quindi non erano utili nè consigliabili a scapito delle prestazioni le funzionalità avanzate proprie dei più recenti linguaggi di programmazione (ad esempio ad oggetti), è stato ritenuto preferibile avvalersi della velocità ed affidabilità caratteristiche del linguaggio C.
C è un linguaggio di programmazione di pregevole essenzialità; la sua semantica utilizza un ristretto insieme di concetti relativamente elementari e simili per natura al funzionamento dell'hardware dei calcolatori; molte istruzioni di C sono direttamente traducibili in una singola istruzione di linguaggio macchina (per esempio, gli operatori di autoincremento e autodecremento).
Nel suddetto linguaggio un ruolo centrale viene svolto dal concetto di puntatore, che viene generalizzato fino a coincidere con l'indirizzamento indiretto: un modo di accedere alla

memoria hardware caratteristico di tutte le moderne CPU. Tale proprietà fa di C un linguaggio particolarmente efficiente.
D'altra parte, rispetto al linguaggio Assembly, C ha in più una struttura logica definita e leggibile, funzioni in stile Pascal e, soprattutto, il controllo sui tipi (in fase di compilazione), completamente assente in Assembly.
Il linguaggio di programmazione C è noto per la sua particolare efficienza, e si è imposto come prodotto di riferimento per la realizzazione di software di sistema su gran parte delle piattaforme hardware moderne. Il processo di standardizzazione del linguaggio (eseguito da due grosse aziende che si occupano di standardizzazione, la statunitense ANSI prima e la svizzera ISO poi) garantisce la portabilità dei programmi scritti in C (spesso detto ANSI C) su qualsiasi piattaforma (in altre parole il software sviluppato con questo linguaggio può essere spostato su qualsiasi macchina senza che si verifichino grossi problemi di esecuzione). Ciò è altresi importante quando si pensa di lavorare allo sviluppo di uno standard.
Oltre che per il software di sistema, C è stato a lungo il linguaggio dominante in una serie di altri domini applicativi caratterizzati da standard di efficienza elevatissimi. Esempi tipici sono le telecomunicazioni, il controllo di processi industriali e il software real-time.
La grammatica e la sintassi del linguaggio C sono particolarmente libere e flessibili, consentendo la scrittura di istruzioni complesse e potenti in solo poche righe di codice (ma anche istruzioni poco leggibili e di difficile comprensione).
Grazie alla particolare efficienza del codice prodotto dai suoi compilatori, il linguaggio C venne impiegato per riscrivere la maggior parte del codice del sistema UNIX, riducendo l'uso dell'Assembly ad un esiguo gruppo di funzioni. Ciò lo rende particolarmente utile nella scrittura di software di sistema, anche per la sua naturale ‘omogeneità’ con sistemi operativi della ‘famiglia’ Unix di cui GNU/Linux utilizzato estensivamente nel corso di tale Dottorato fa parte.
Grazie alla scelta di questo linguaggio è stato possibile avvalersi della libreria standard del linguaggio C e di una libreria aggiuntiva molto ben scritta e completa già compresa in PostgreSQL: libpq.

4.10.2 Libreria aggiuntiva Postgres: libpq-fe.h

Oltre alle funzioni di libreria standard presenti nel C è stata utilizzata per interfacciarsi a PostgreSQL la sua libreria nativa di interfaccia, ovvero Libpq, di cui nel seguito di questo

paragrafo vengono descritte alcune caratteristiche importanti allo scopo di far rilevare come la scelta di software Open Source abbia come conseguenza la immediata disponibilità di strumenti gratuiti e ad alto livello su cui basare lo sviluppo dei propri progetti. Al solito questa caratteristica è molto utile nello sviluppo di uno standard.
Libpq è un insieme di funzioni che consentono a programmi client di far eseguire ad un server PostgreSQL diverse operazioni e di riceverne i risultati. Tali funzioni costituiscono il motore per le applicazioni scritte in vari linguaggi (C, Perl, Python…) che dialogano con il RDBMS.
Un programma applicativo può mantenere diverse connessioni attive contemporaneamente (ad esempio per poter in tal modo gestire l'accesso a più di un database). Ogni connessione è gestita da un oggetto PGconn, restituito dalla funzione PQconnectdb o PQsetdbLogin.
È importante notare come queste funzioni restituiscano sempre un puntatore non nullo ad un oggetto, a meno che non vi sia abbastanza memoria da destinare all'oggetto PGconn. Per verificare se una connessione è stata effettuata con successo è possibile invocare la funzione PQstatus prima di inviare una query tramite l'oggetto connection.
Di seguito verranno descritte alcune delle funzioni utilizzate per la connessione ad un server di backend PostgreSQL.

PQconnectdb

Per realizzare una nuova connessione al database server è necessario usare la sintassi:
PGconn *PQconnectdb(const char *conninfo).

Questa funzione apre una nuova connessione al database usando i parametri passati con la stringa conninfo. Diversamente da PQsetdbLogin, l'insieme dei parametri può essere esteso senza modifiche alla function signature, cosicchè l'uso di questa funzione (o delle sue analoghe non bloccanti PQconnectStart e PQconnectPoll) sarà preferibile nella creazione di nuove applicazioni. La stringa passata può essere vuota, per utilizzare tutti i parametri di default, oppure può contenere l'impostazione di uno o più parametri, separati da uno spazio. L'impostazione di ogni parametro è nella forma parola_chiave = valore. Gli spazi prima e/o dopo il segno di uguale sono opzionali. Per scrivere un valore vuoto, oppure un valore che contiene degli spazi, bisogna comprenderlo tra singoli apici, ad esempio, parola_chiave = 'un valore'. Tra le parole chiave attualmente riconosciute come parametri vi sono le seguenti.

- Host: indica il nome dell'host a cui connettersi. Se inizia con una barra specifica una comunicazione con un dominio Unix piuttosto che una comunicazione TCP/IP; il valore

rappresenta il nome della directory dove è memorizzato il file socket. Il comportamento predefinito quando l'host non è specificato è di connettersi ad un socket di dominio Unix in /tmp (o a qualsiasi altra directory di socket specificata durante la compilazione di PostgreSQL). In una macchina senza socket di dominio Unix, l'impostazione predefinita è connettersi a localhost.

- Hostaddr: è l'indirizzo numerico IP dell'host a cui connettersi. Deve essere nel formato standard IPv4,ad esempio, 172.28.40.9. Se la macchina li supporta, è possibile anche usare gli indirizzi IPv6. Qualora non venga specificato alcun parametro, viene utilizzata di default la comunicazione TCP/IP. Usando hostaddr invece di host evitiamo alle applicazioni la risoluzione del nome di host, che potrebbe essere importante per quelle applicazioni con vincoli sui tempi di esecuzione. Tuttavia, le autenticazioni Kerberos e GSSAPI richiedono il nome di host. Senza un nome host o un indirizzo host, libpq si connette utilizzando socket di dominio Unix locale, oppure su una macchina senza socket di dominio Unix, cercherà di connettersi a localhost.
- Port: indica il numero della porta di connessione al server host, oppure l'estensione del nome del socket file per la connessione ad un dominio Unix.
- Dbname: è il nome del database. Il valore predefinito è lo stesso nome dell'utente.
- User: riferisce al nome dell'utente con cui ci si connette a PostgreSQL. Il valore predefinito è lo stesso dell'utente del sistema operativo sul quale viene eseguita l'applicazione.
- Password: è la password da utilizzare se richiesta dal server per l'autenticazione.
- Connect_timeout: specifica il tempo massimo di attesa per la connessione espresso in secondi (scritto come un intero decimale in formato stringa). Zero o non specificato significa aspettare all'infinito. Non è raccomandato utilizzare un timeout inferiore a 2 secondi.
- Options:contiene opzioni inviate al server da linea di comando.
- Sslmode: determina se e con quale priorità sarà negoziata una connessione SSL con il server. Ci sono quattro modalità: disable tenterà solo una connessione SSL non cryptata; allow proverà per prima una connessione non-SSL, in caso di fallimento, tenterà una connessione SSL; prefer (opzione di default) proverà prima una connessione SSL, in caso di fallimento, tenterà di utilizzare una regolare connessione non-SSL; require tenterà solo una connessione SSL. Se PostgreSQL è stato compilato senza il supporto SSL, l'uso dell'opzione require causerà un errore, mentre le opzioni allow e prefer saranno accettate, ma libpq di fatto non tenterà una connessione SSL.

- Krbsrvname: è il nome del servizio Kerberos da utilizzare per l'autenticazione con Kerberos 5 o GSSAPI. Per riuscire, deve coincidere con il nome del servizio specificato nella configurazione del server.
- Gsslib: indica la libreria GSS da utilizzare per l'autenticazione GSSAPI. Utilizzata solo in Windows. Impostare a gssapi per forzare libpq ad utilizzare, per l'autenticazione, la libreria GSSAPI invece del default SSPI.

PQconnectStart e PQconnectPoll

Vengono utilizzate per realizzare una connessione al database server in modalità non bloccante. La sintassi da utilizzare è

PGconn *PQconnectStart(const char *conninfo)

per la prima e

PostgresPollingStatusType PQconnectPoll(PGconn *conn)

per la seconda.

Tramite queste due funzioni utilizzate è possibile aprire una connessione con il database server, tale che l'esecuzione dei processi dell'applicazione che apre la chiamata non siano bloccati da una attività remota di I/O.

Lo scopo di questo approccio è poter gestire l'attesa conseguente al completamento di una operazione di I/O all'interno del loop principale dell'applicazione, piuttosto che delegarlo all'interno di PQconnectdb, in modo che l'applicazione possa gestire questa operazione in parallelo ad altre attività.

PQstatus

Nel corso di una connessione, lo stato della stessa può essere verificato con una chiamata a questa funzione. Se viene ritornato CONNECTION_BAD, allora la procedura di connessione è fallita; se viene ritornato CONNECTION_OK, allora la connessione è stata realizzata.

Nel caso di procedura di connessione asincrona è possibile che si verifichino altri stati. Questi indicano la fase corrente di una procedura di connessione e possono essere utilizzati, ad esempio, per fornire informazioni all'utente.

Gli stati contemplati sono:

- CONNECTION_STARTED indica l'attesa di completamento della connessione
- CONNECTION_MADE indica che la connessione è stata eseguita

- CONNECTION_AWAITING_RESPONSE quando la connessione è in attesa di risposta da parte del server
- CONNECTION_AUTH_OK quando l'autenticazione è avvenuta correttamente e si attende la conclusione del processo di connessione
- CONNECTION_SSL_STARTUP quando si sta negoziando una sessione criptata SSL
- CONNECTION_SETENV quando si sta negoziando l'impostazione di alcuni parametri detti environment-driven.

PQfinish

chiude le connessioni al server e libera la memoria utilizzata dall'oggetto PGconn. La sintassi da utilizzare è:

void PQfinish(PGconn *conn).

È importante ricordare che, anche se la connessione fallisce (come indicato da PQstatus), l'applicazione deve chiamare PQfinish per liberare la memoria utilizzata dall'oggetto PGconn. Dopo aver chiamato PQfinisch il puntatore PGconn non può più essere utilizzato.

PQreset

Reimposta il canale di comunicazione con il server. La sintassi per utilizzare questo comando è:

void PQreset(PGconn *conn).

Questa funzione chiude la connessione al server e cerca di ristabilirne una nuova usando i parametri utilizzati in precedenza. Questa funzione potrebbe essere utile per recuperare una connessione persa a causa di un errore mentre era in attività.

PQexec

Sottopone un commando al server e attende la risposta. Per eseguirlo è necessario usare la seguente sintassi:

PGresult *PQexec(PGconn *conn, const char *command)

L'esito del comando è generalmente un puntatore non nullo (sarà nullo soltanto nel caso in cui si abbia un errore grave o il server esaurisca la memoria a sua disposizione) .

È possibile includere più comandi SQL separati da un punto e virgola nella stessa stringa di comando. In questo caso sarà possibile vedere soltanto l'esito dell'ultimo comando inviato oppure, qualora uno dei comandi desse esito negativo, l'esecuzione si fermerebbe e si avrebbe come esito il messaggio dell'errore avvenuto.

PQexecParams

Sottopone un comando al server con la possibilità di specificare dei parametri e attende la risposta. La sintassi da utilizzare è la seguente:

PGresult *PQexecParams(PGconn *conn, const char *command, int nParams, const Oid *paramTypes, const char * const *paramValues, const int *paramLengths, const int *paramFormats, int resultFormat)

PQexecParams è simile a Pqexec, ma offre come funzionalità aggiuntiva quella di poter specificare dei parametri separatamente rispetto alla stringa di comando. I risultati possono -in aggiunta- essere espressi sia in formato testuale che binario.

Gli argomenti passabili come parametro sono quindi:

- conn:che indica la connessione sulla quale passare il comando
- command: che indica la stringa di comando da eseguire (se vengono utilizzati dei parametri si può riferire a loro indicando il carattere $ seguito dal numero progressive del parametro: $1, $2 ecc.)
- nParams: indica il numero dei parametri passati alla funzione ed è anche la lunghezza degli array paramTypes[], paramValues[], paramLeghts[] e paramFormats[]
- paramTypes[]: specifica il tipo di dato assegnato ad ogni parametro
- paramValues[]: specifica il valore di ogni parametro
- paramLenghts[]: specifica la dimensione di ogni parametro
- paramFormats[]: specifica se il formato dei parametri è testuale o binario
- resultFormat: specifica se il risultato va ottenuto in formato binario o formato testuale (si usa il valore 1 per indicare il formato testuale e 0 per indicare il formato binario)

Diversamente da PQexec, PQexecParams permette di inserire soltanto un comando SQL alla volta, ma ha il grande vantaggio di consentire di separare i parametri dalla linea di comando SQL.

PQresultStatus

Restituisce il risultato di un comando. La sintassi da utilizzare è:

ExecStatusType PQresultStatus(const PGresult *res).

I valori restituiti da questo comando possono essere:

- PGRES_EMPTY_QUERY qualora la stringa inviata al server fosse vuota
- PGRES_COMMAND_OK qualora il comando sia stato eseguito correttamente e non ci fosse alcun valore di ritorno
- PGRES_TUPLES_OK qualora il comando sia stato eseguito correttamente e ci siano dei valori di ritorno
- PGRES_COPY_OUT qualora sia iniziato un trasferimento di dati dal server
- PGRES_COPY_IN qualora sia iniziato il trasferimento di dati nel server
- PGRES_BAD_RESPONSE qualora la risposta del server non sia stata compresa (accade quando c'è un errore nel server)
- PGRES_NONFATAL_ERROR quando si è rilevato un errore non grave (una notifica o un avvertimento)
- PGRES_FATAL_ERROR quando si è rilevato un errore grave;

Se dalla funzione PQresultStatus provenisse il risultato PGRES_TUPLES_OK, si potrebbero utilizzare le funzioni sotto riportate per estrarre i dati risultanti dalla query.

PQntuples

Restituisce il numero di righe (tuple) del risultato della query come numero intero. La sintassi da utilizzare è:

int PQntuples(const PGresult *res)

PQnfields

Restituisce il numero di colonne (campi) in ogni riga del risultato della query. La sintassi da utilizzare è:

int PQnfields(const PGresult *res)

PQfname

Restituisce il nome della colonna associato al numero passato come parametro. La numerazione delle colonne parte da 0. La sintassi da utilizzare è:
char *PQfname(const PGresult *res, int column_number)

PQfnumber
Restituisce il numero della colonna associata al nome passato per parametro. Qualora non ci fosse alcuna colonna che risponda al nome passato come parametro, la funzione restituisce il numero -1. La sintassi da utilizzare è:
int PQfnumber(const PGresult *res, const char *column_name)

Esistono ancora diverse altre funzioni tramite le quali è possibile interrogare il database, ottenere dati contenuti al suo interno, e modificarlo persino nella sua struttura.
È dunque possibile, tramite questo strumento, interagire pienamente con il database, senza necessità di programmare a basso livello.

4.10.3 Applicativo realizzato

Tramite questo software è possibile eseguire tutte le operazioni necessarie per l'interazione con il database e previste nella Proof of Concept. Ad esempio è possibile inserire un nuovo tipo di terminale nell'elenco dei tipi le cui istanze diventano dinamicamente inizializzabili, oppure inizializzare direttamente una istanza. Possiamo considerarlo dunque un prototipo che verrà usato personalizzandolo ed implementandolo a seconda delle differenti necessità. Nel paragrafo 6.6 invece verrà presentato il software sviluppato per un terminale effettivamente progettato e realizzato che rileva dati da una periferica GPS, e li inserisce in database. Il software sul terminale effettua una interazione complessa con il prototipo del sistema di fleet management via Internet, nel seguito ne verranno descritte le funzionalità. In questo paragrafo verrà descritto il funzionamento del solo software di gestione Database.
La prima parte del codice comprende la serie tipica di comandi per poter usufruire di funzioni definite nelle librerie standard C e nella libreria libpd oggetto del precedente paragrafo.
In seguito è possibile trovare due funzioni: "exit_nicely" e "write_error".
La prima delle due consente di uscire correttamente dal programma.
Quando si interagisce con PostgreSQL, infatti, viene aperta una connessione con il database e per questa connessione viene allocata una parte di memoria. Per uscire correttamente da un

programma in esecuzione che abbia creato una connessione con un database, è necessario chiudere questa connessione per liberare la memoria occupata. Nel codice, al momento di dover terminare l'esecuzione, si richiama perciò la funzione "exit_nicely" che esegue il comando "PQfinish" per chiudere la connessione, ed il comando "exit" per terminare l'esecuzione.
La funzione "write_error" invece consente di scrivere gli errori che dovessero sopravvenire in un file chiamato "error_log".
I parametri accettati in ingresso dalla funzione sono: temp_str, conn, input.
Il primo di questi contiene l'errore sopravvenuto, il secondo le informazioni sulla connessione al database, mentre il terzo include la riga di comando che ha chiamato in esecuzione il codice.
La funzione si occupa di scrivere sul file "error_log" l'errore sopravvenuto, l'orario in cui si sia verificato e la stringa che, chiamando in esecuzione il codice, lo ha generato.
In seguito a queste funzioni inizia il programma principale.
Dopo una parte di dichiarazione di variabili, viene creata una stringa che riproduce il comando che ha generato l'esecuzione del codice (che sarà utilizzata per essere passata come parametro alla funzione "write_error", qualora sopravvenga un errore), si effettua la connessione al database, e viene compiuto un controllo sulla connessione. Se tale controllo non dà esito positivo, è opportunamente segnalato in una variabile il tipo di errore verificatosi, e si esegue la chiamata alla funzione per la scrittura degli errori.
In seguito vi è un controllo sul primo parametro di chiamata: il programma utilizza infatti questo parametro per discriminare le operazioni da eseguire.
Le opzioni previste per questo parametro sono:
- 1 per l'inserimento di dati
- 99 per l'inserimento di un nuovo tipo nell'elenco di tipi inizializzabili
- 999 per l'inizializzazione di una nuova periferica

Per l'inserimento di dati, il programma deve essere lanciato rispettando la seguente sintassi:
./mis 1 tipo_periferica seriale_periferica dati_da_inserire.

Vengono quindi eseguiti i seguenti passaggi:
- un controllo sul numero di parametri inseriti (se non sono almeno cinque comprensivi del nome del programma e del parametro "1" c'è sicuramente un errore poiché i primi quattro

servono solo per chiamare il programma ed identificare la sorgente dei dati, dal quinto parametro in poi invece si trovano i dati da inserire
- con il comando "PQexecParams" viene effettuata una ricerca nel database per trovare la periferica specificata tramite il tipo e il seriale (se non viene trovata, significa che la periferica non è ancora stata inizializzata)
- si effettua un altro controllo sul numero di parametri (una volta identificata la periferica è infatti possibile sapere quanti sono i dati che devono essere inviati da dispositivi di quel tipo)
- si tenta l'inserimento dei dati preparando in una stringa la query da effettuare ed eseguendola tramite il comando "PQexec", qualora questo vada a buon fine il programma può terminare, altrimenti si suppone vi sia un errore nel tipo di dati inviati, per cui viene chiamata di nuovo la funzione di scrittura errori.

Se in alcuno dei passaggi sopra indicati si rileva un errore, oppure uno dei controlli non dà esito positivo, viene annotato l'errore in una variabile temporanea e, passando come parametri questa variabile, i dati sulla connessione, e la stringa contenente il comando che ha generato l'esecuzione del codice, si chiama la funzione "write_error" e si compilano alcune righe nel file "error.log".

Per l'inserimento di un nuovo tipo di periferiche, il programma prevede che il comando di chiamata rispetti la seguente sintassi:
./mis 99 tipo_periferica istruzioni_inizializzazione.

Vengono quindi eseguiti i seguenti passaggi:
- un controllo sul numero di parametri inseriti (se non sono almeno quattro, comprensivi del nome del programma e del parametro "99", c'è sicuramente un errore)
- si verifica che il tipo della periferica compaia nell'elenco dei tipi inizializzabili (se compare, infatti, non è necessario inserirlo di nuovo)
- si crea una stringa contenente il comando per creare una tabella, avente come nome il tipo di periferica, e come attributi il contenuto delle "istruzioni_inizializzazione"
- si esegue la query di creazione tabella
- tramite il comando PQexecParams si inserisce il tipo nella lista dei tipi inizializzabili.

Se in alcuno dei passaggi sopraindicati si rileva un errore, o uno dei controlli non dà esito positivo, viene annotato l'errore in una variabile temporanea e, passando come parametri questa variabile, i dati sulla connessione, e la stringa contenente il comando che ha generato l'esecuzione del codice, si chiama la funzione "write_error" e si scrivono alcune righe nel file "error.log".

Per l'inizializzazione di una nuova periferica, il programma prevede la seguente sintassi:
./mis 999 tipo_periferica seriale periferica

Vengono quindi eseguiti i seguenti passaggi:
- un controllo sul numero di parametri inseriti (se non sono almeno quattro comprensivi del nome del programma e del parametro "999" c'è sicuramente un errore)
- tramite il comando "PQexecParams" si controlla se il tipo della periferica compare nella lista dei tipi inizializzabili
- si esegue una query per controllare che la periferica non sia stata già precedentemente inizializzata
- si tenta l'inserimento dei dati della periferica tra quelle inizializzate.

Se in qualcuno dei passaggi sopra indicati si rileva un errore, o uno dei controlli non dà esito positivo, viene annotato l'errore in una variabile temporanea e, passando come parametri questa variabile, i dati sulla connessione e la stringa contenente il comando che ha generato l'esecuzione del codice, si chiama la funzione "write_error" e si scrivono alcune righe nel file "error.log". Infine, qualora il primo parametro non contenga una delle opzioni contemplate (1, 99 o 999) si annota l'errore nel file "error.log".

4.11 Proof of Concept - Terminale progettato e realizzato

In questo paragrafo viene descritto il prototipo di terminale realizzato per la Proof of Concept del task 0.
Il prototipo è fisicamente costituito da:
- una periferica GPS dotata di antenna esterna
- una scheda x86 compatibile in standard PC/104 che fa girare il software applicativo

La periferica utilizzando la sua antenna riceve dati GPS, e tramite la porta seriale li rende disponibili alla scheda PC/104 usando il protocollo NMEA. La scheda PC/104, utilizzando un software Open Source che gestisce la periferica via seriale in combinazione con l'applicativo sviluppato ad hoc, è in grado di interrogare i dati resi disponibili dal sottosistema GPS e di inserirli nel database MIS. Combinando le funzionalità del software di gestione presentato nel paragrafo 6.3 inteso per girare sul MI Server e del software applicativo sviluppato per il terminale, è possibile realizzare tutte le funzionalità previste nella Proof of Concept sfruttando le funzionalità offerte dalle reti standard TCP/IP.
Di seguito verranno descritte nel dettaglio le tecnologie impiegate.

4.11.1 Periferica GPS

La periferica utilizzata per la realizzazione del prototipo è il GARMIN GPSMAP® 76S visibile in Figura 4.7.
Tale apparecchio è caratterizzato da una notevole robustezza oltre ad essere impermeabile a norma IPX e galleggiante, è dotato di batterie che ne garantiscono il funzionamento lontano da fonti di alimentazione per molte ore, ha la possibilità di utilizzare una antenna esterna che migliora decisamente la ricezione dei segnali GPS ed inoltre può essere collegato tramite porta seriale in standard RS-232 per esportare dati ad altre apparecchiature.

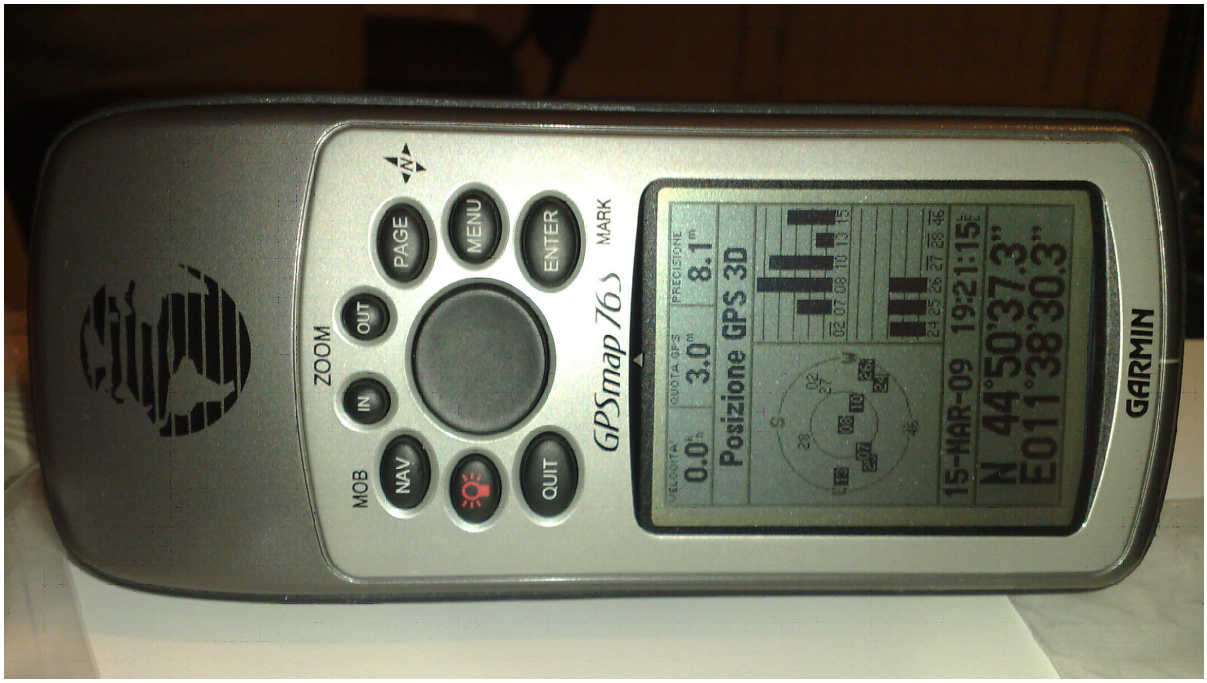


Figura 4.7: Garmin GPSMAP 76S

Il Garmin GPSMAP 76S utilizza il protocollo NMEA per trasmettere dati sulla porta seriale. NMEA 0183 (o più comunemente NMEA) è uno standard sia elettrico che di comunicazione di dati utilizzato soprattutto in nautica per la comunicazione tra dispositivi come sonar, anemometri, rilevatori di eco e piloti automatici, e molto spesso nella comunicazione di dati satellitari GPS.

L'ente che gestisce e sviluppa il protocollo è la statunitense National Marine Electronics Association. Lo standard utilizza un semplice protocollo di comunicazione seriale ASCII che si basa sul principio che la fonte dei dati, denominata 'talker', può soltanto inviare dati 'sentences' a riceventi multipli, definiti 'listener', che possono esclusivamente riceverli.

Tutte le sentence inviate tramite questo protocollo sono formate da una parte iniziale (detta prefisso) che contiene le informazioni sull'apparecchiatura che invia i dati, una parte centrale che contiene i dati veri e propri, e una parte finale che contiene il checksum (due caratteri che formano un numero esadecimale calcolato sulla sentence inviata e che permette di controllarne l'integrità). In bibliografia sono presenti alcuni riferimenti al protocollo NMEA.

4.11.2 Lo standard PC/104

Si introduce qui un'altro degli standard aperti utilizzati nel lavoro di Dottorato, in questo caso rivolto all'hardware. PC/104 è uno standard per computer embedded (computer deputati all'esecuzione di una o più funzioni dedicate) di piccole dimensioni (small form) controllato dal PC/104 Consortium, che ne definisce sia le specifiche che il bus. Il PC104 Consortium è costituito da membri in tutto il mondo ed ha il compito di definire e mantenere gli standard e diffondere informazioni circa gli standard PC/104 ed altri riguardanti macchine small form factor tra i costruttori ed utenti. Inoltre il consorzio si occupa di agevolare i collegamenti ed il dialogo tra la comunità che fa riferimento allo standard aperto PC/104 ed altre organizzazioni che si occupano di standardizzazione.
Oltre allo standard PC/104 il Consorzio gestisce anche altri standard hardware come PC/104-Plus, PCI-104, EBX and EPIC. Tutti questi standard sono liberamente disponibili, ed utilizzano moduli di piccole dimensioni (3.6 x 3.8 pollici) ed impilabili, con connettori di espansione con bus ISA 104-pin oppure PCI 120-pin per I/O. La tecnologia di impilamento consente di collegare ad un sistema moduli multipli senza complicazioni di backplane o altri

tipi di contenitori. Infatti diversamente dal popolare ATX (che utilizza il bus PCI comunemente utilizzato in tutti i Personal Computer), l'architettura che compone un PC/104 non ha un backplane (bus che interconnette i diversi moduli di comunicazione all'interno di un dispositivo), e questo consente a più moduli PC/104 di essere 'impilati' creando dei 'blocchi', e permettendo così di costituire architetture articolate e allo stesso tempo molto solide e di ridotte dimensioni. Il grandissimo vantaggio dello standard PC/104 e PC/104plus è che in piccole dimensioni ed a costi accettabili sono rese reperibili piattaforme hardware adatte a realizzazioni di tipo embedded ma che mantengono una compatibilità pressoché assoluta con il mondo PC x86, consentendo l'utilizzo di S.O. e strumenti software standard. In bibliografia sono presenti riferimenti per reperire ulteriori informazioni. E' di tutta evidenza come l'utilizzo di hardware conforme ad uno standard aperto è assolutamente in sintonia con gli obiettivi del presente Dottorato, e consente di ottenere tecnologie di alto livello a basso costo e svincolandosi anche per la parte hardware da un particolare produttore. Ciò consente inoltre insieme a software Open Source di assemblare piattaforme complete di hardware, software di base, framework di sviluppo e applicativo completamente aperte e standard per le proprie applicazioni. Le piattaforme PC/104 e PC/104plus verranno utilizzate anche nei Task successivi, scegliendo tra le varie soluzioni disponibili in base alle esigenze della particolare implementazione. Nel paragrafo successivo è introdotta la prima piattaforma utilizzata, una scheda PC/104 di prestazioni e consumo ridotti adatta alla realizzazione di un singolo terminale dati. Sono ormai disponibili piattaforme standard PC/104 con rilevati potenze di calcolo, ed a breve si attende l'introduzione di soluzioni con supporto multi-core.

4.11.3 La scheda in standard PC/104

Nel caso del Proof of Concept del Task 0 si è deciso di utilizzare una scheda compatibile con lo standard PC/104 prodotta da Icop Technology Inc. modello Vortex 6071LV.
Alcune specifche del modulo sono di seguito elencate per dare una idea del tipo di piattaforma utilizzata:

- Vortex86™ System-on-Chip CPU–200MHz
- RAM Onboard 128MB SDRAM
- Bus PC/104 Standard
- Interfaccia I/O Enhanced IDE
- tre porte RS232

- una porta RS232/485
- una porta Parallela
- Interfaccia FDD
- Due porte USB (Ver. 1.1)
- Ingressi e uscite audio
- Scheda video VGA AGP Rev.2.0 con memoria video Shared fino a 16MB.
- Risoluzione fino a 1,920x 1,440 true color. Collegabile a display CRT/LCD
- LAN Realtek 8100B 10/100Mbps
- Flash Disk ATA-Disk-Module da 2GB
- Alimentazione: Singola tensione +5V @1.0 A
- Dimensioni scheda 90mm x 96mm
- Peso scheda 110g
- Temperatura operativa -20°C ~ +70°C

Nella seguente Figura 4.8 è ritratta la scheda montata all'interno di un contenitore:

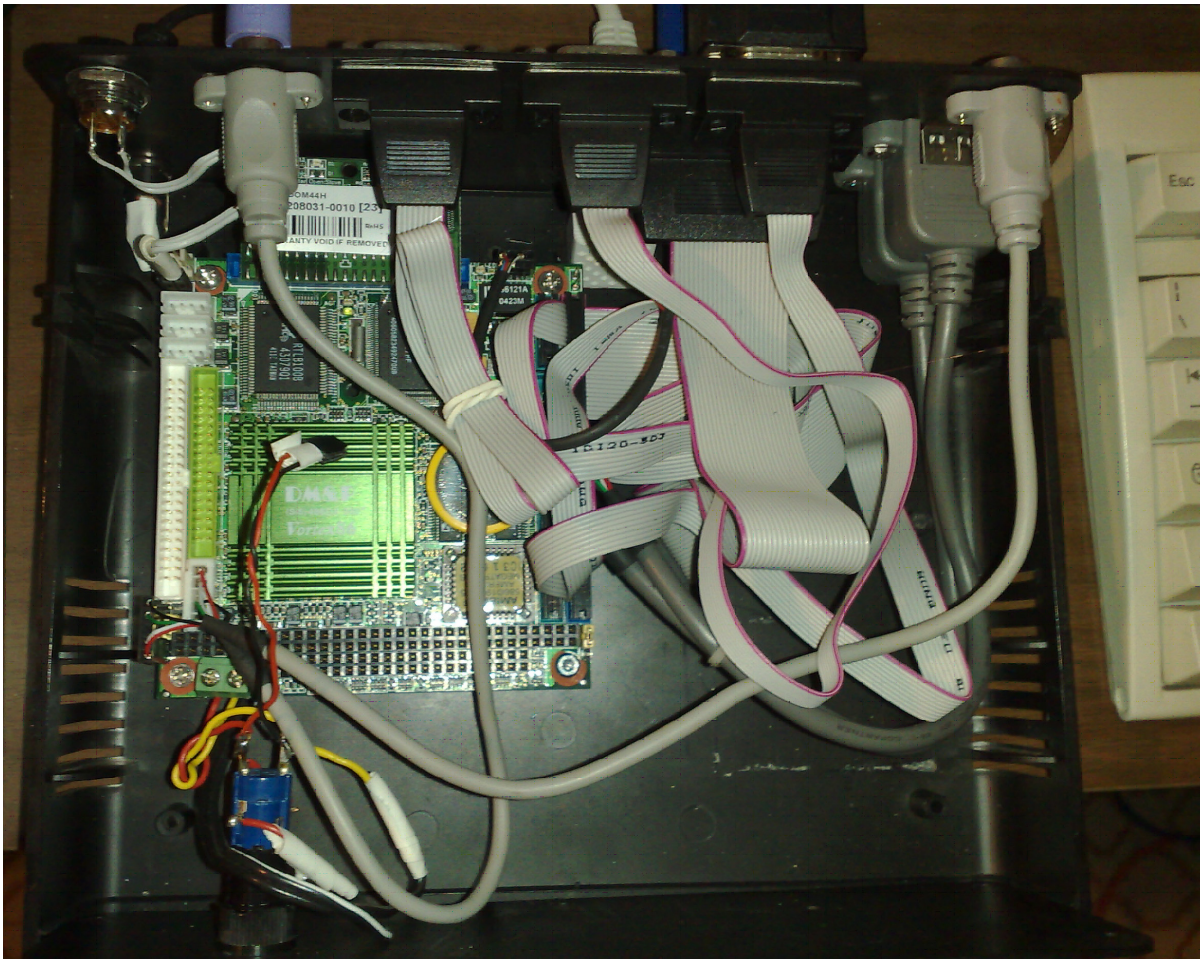


Figura 4.8: Scheda PC104 Icop Technology Inc. montata in contenitore

Queste schede permettono quindi di avere in dimensioni molto ridotte un vero e proprio PC. Per la realizzazione del terminale dati è stato utilizzato un S.O. Linux CentOS 4. Viste le prestazioni limitate dell'hardware si è preferito optare per una versione precedente adatta per sistemi a basse prestazioni. Il sistema operativo è stato configurato in modo essere il più leggero possibile disattivando di default l'interfaccia grafica e tutti i servizi non necessari, in modo da riuscire ad ottenere prestazioni soddisfacenti. Per lo sviluppo delle applicazioni è stata scelta la strada di virtualizzare una installazione di CentOS 4 all'interno del server di sviluppo dotato di CentOS 5, in modo da semplificare e rendere più rapido lo sviluppo stesso. Come software applicativo sono stati utilizzati una applicazione sviluppata ad hoc ed il software Open Source Gpsd di cui si parlerà nel paragrafo successivo.

4.11.4 Il software GPSD

Gpsd è un 'demone' ovvero un servizio del sistema operativo che riceve dati da una periferica GPS, e li restituisce anche a multiple applicazioni, di fatto virtualizzando la periferica stessa, che tramite connessione diretta via RS-232 potrebbe dialogare con una singola applicazione alla volta. Gpsd è usato comunemente su sistemi Linux e FreeBSD, ed è un software . Il demone fornisce i suoi servizi tramite protocollo TCP/IP operando il binding alla porta 2947. Accetta comandi leggendoli da tale socket, e tramite la stessa via ritorna i suoi risultati. I comandi nelle versioni più recenti adottano una sintassi basata su JavaScript Object Notation (JSON) e ritornano risposte con lo stesso standard (versioni precedenti, come quella adottata in questo caso, usano comandi più semplici basati su di una singola lettera). La maggior parte dei ricevitori GPS è supportata, sia con connessione seriale, che USB o Bluetooth. Versioni recenti di Gpsd supportano anche ricevitori Automatic Identification System (AIS) che è un sistema di tracciamento automatico usato in ambito navale e basato sullo scambio reciproco di informazioni tra vascelli. Opzionalmente Gpsd può interfacciarsi via con il servizio UNIX ntpd che implementa il network time protocol via shared memory in modo da sincronizzare data ed ora di sistema usando direttamente l'orologio GPS.Tramite questo software è dunque possibile consentire ad un dispositivo GPS di essere utilizzato da più processi, o da più utenti, senza pericolo di interferenze o di perdita di dati.

Il grande vantaggio di questo servizio è che esso risponde alle richieste dati in un modo assai più semplice ed intuitivo rispetto a quanto definito nello standard NMEA. È dunque molto più

semplice utilizzare i dati una volta prelevati dalla socket, ed il lavoro di scrittura di una applicazione che necessiti interagire con una periferica GPS risulta decisamente semplificato.
La modalità di interrogazione di questo sistema è molto semplice: ogni richiesta può infatti essere inoltrata semplicemente tramite un carattere ASCII.
Di seguito vengono riportate alcune delle richieste che è possibile fare a gpsd per dare una idea delle funzionalità offerte dal software:

- a: restituisce l'altitudine corrente espressa in metri sul livello del mare
- d: restituisce l'orario UTC nel formato ISO 8601 "D=yyyy-mm-ddThh:nmm:ss.ssZ"
- f: restituisce o stabilisce il nome del dispositivo GPS attivo. Se viene eseguito da solo restituisce una stringa del tipo "F=" seguita dal nome del dispositivo attivo, altrimenti è possibile eseguire tale richiesta inserendo "f=" seguito da una serie di caratteri che verranno considerati come il nome di un terminale GPS (se il terminale è nella lista dei terminali viene aperto, si controlla che il terminale sia effettivamente collegato, e qualora lo sia, il terminale diventa il terminale attivo)
- m: restituisce diversi valori a seconda del modo in cui è possibile ricevere i dati dalla periferica, 0 per indicare che non è stato stabilito un modo per ricevere dati, 1 per indicare che non è possibile ricevere dati, 2 per indicare che è possibile ricevere dati bidimensionali (senza altitudine), 3 per indicare che è possibile ricevere dati tridimensionali
- p: restituisce la posizione corrente indicando per prima la latitudine e esprimendo i valori in gradi
- r: seleziona o deseleziona la modalità "RAW"
- w: seleziona o deseleziona la modalità "watcher"
- x: restituisce uno 0 se il dispositivo GPS è disattivato, altrimenti restituisce una stringa del tipo "X=" seguito dal "timestamp" dell'ultima volta in cui si è ricevuto un dato

Riferimenti per reperire altre informazioni su Gpsd sono presenti in bibliografia.

4.11.5 Software applicativo del terminale

Il codice sviluppato consente di simulare il comportamento di un terminale a bordo di un veicolo dedicato al rilevamento e alla storicizzazione di dati di posizione rilevati tramite periferica GPS. Le funzioni supportate sono:

- inizializzazione di un nuovo tipo di terminale all'interno del database MI System, con associati al tipo una struttura dati creata dinamicamente destinata a contenere il tipo di dati che verrà effettivamente trasmesso.
- inizializzazione di una istanza di terminale di un tipo riconosciuto. Tale operazione è preliminare all'invio di dati vero e proprio e consente di associare i dati poi trasmessi al singolo terminale della flotta
- invio di una sequenza di dati di posizione rilevati tramite GPS

Le funzioni dimostrano la totale dinamicità del sistema immaginato, per cui una volta abilitata dal software di gestione una categoria di terminali all'accesso è il terminale stesso che può autoinizializzarsi e fornire dinamicamente la struttura dei dati che saranno inviati, e successivamente inviarli in maniera automatica.

Nel seguito è descritta la implementazione del software realizzato.

Nella prima parte del codice troviamo le inclusioni delle stesse librerie che sono state utilizzate nel prototipo di software di gestione del Database. In seguito vi è la definizione di numerosi parametri utili per l'esecuzione del software, ad esempio per apertura della socket di comunicazione con Gpsd e per l'interazione con il database PostgreSQL.

Si passa quindi alla definizione di alcune funzioni di seguito elencate per meglio chiarire alcune funzionalità:

- gpsd_talk_row: riceve informazioni dalla socket 'gestita' da Gpsd
- gpsd_connect: connesione alla socket 'gestita' da Gpsd
- gpsd_init: inizializzazione della connessione vera e propria e verifica del fix GPS
- gps_get_timestamp: richiede a gpsd data e ora dell'ultimo ricevimento di dati validi dalla periferica GPS
- gps_get_position: interrogare la periferica GPS sulla posizione corrente e la salva
- write_log: scrive sui log, in questo software infatti sono previsti un log per gli errori ed un altro log per le comunicazioni di normale funzionamento del software
- exit_nicely: chiude correttamente la connessione al database nel caso in cui il programma debba essere terminato
- db_init: inizializza la connessione al database PostgreSQL

In seguito si ha il programma principale, per il quale è stata scelta una modalità di funzionamento leggermente differente rispetto al software di gestione database per consentire

all'utente di selezionare quale operazione eseguire. In questo caso infatti si è scelto di operare sul numero di parametri inseriti dall'utente:

- nessun parametro: non vi è nessuna interazione con il database, il software si limita a stampare a video una sequenza di dati ricavati dalla periferica GPS
- un parametro: vengono scritte sul database dieci rilevazioni consecutive di posizione GPS associandole nel database terminale il cui ID è passato come parametro
- due parametri: viene inizializzato un nuovo terminale che ha come tipo e seriale i due parametri passati
- tre parametri (di cui il primo deve essere la parola "new"): viene inizializzato un nuovo tipo di terminale con nome e formato della struttura dati da associare passati come parametro

Dunque all'avvio un primo controllo sul numero di parametri e si prepara la stringa di chiamata del software, per inserirla eventualmente nel file di log, qualora sopravvenga un errore; si inizializzano a seconda dei parametri passati il database, il GPS o entrambi, e a seconda del numero di parametri passati si procede con operazioni diverse.

Nel caso in cui non vi sia alcun parametro, viene inizializzata soltanto la connessione con Gpsd, in quanto non vi sarà interazione con il database, viene verificato se la periferica GPS è in grado di fornire una posizione, e si stampano infine a video dieci posizioni consecutive ricevute dalla periferica stessa.

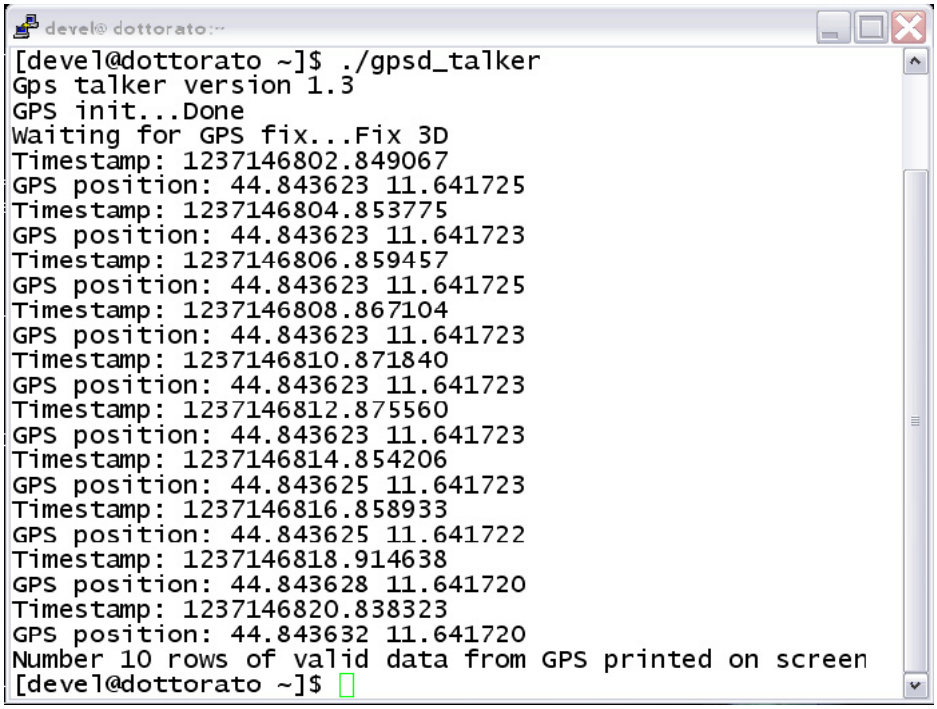


Figura 4.9: Terminale GPS: stampa dati a video

In Figura 4.9 è presentato l'output a video nel caso appena descritto. Osservando il timestamp si nota che il tempo di attesa tra un dato e l'altro è impostato a due secondi, e ciò è stato fatto perché il periodo di update del GPS è pari a un secondo, ed è necessario quindi attendere un tempo più lungo per essere sicuri di ottenere dati aggiornati.
Dallo screenshot si può anche rilevare la modalità con cui si controlla se l'ultimo dato disponibile è aggiornato, infatti normalmente gpsd rende disponibile sempre lo stesso dato finché non ne riceve uno nuovo. È possibile però richiedere al demone un timestamp, cioè una stringa che contiene data e ora di ricevimento dell'ultimo dato che è disponibile. Salvando ogni volta questa informazione è possibile confrontarla con quella che abbiamo ricevuto la volta precedente, e capire se il dato presente via socket è aggiornato e quindi la periferica non è stata spostata, oppure è lo stesso già rilevato ad esempio a causa della perdita del fix GPS.

In Figura 4.10 è possibile vedere la risposta del sistema ad una chiamata con un parametro:

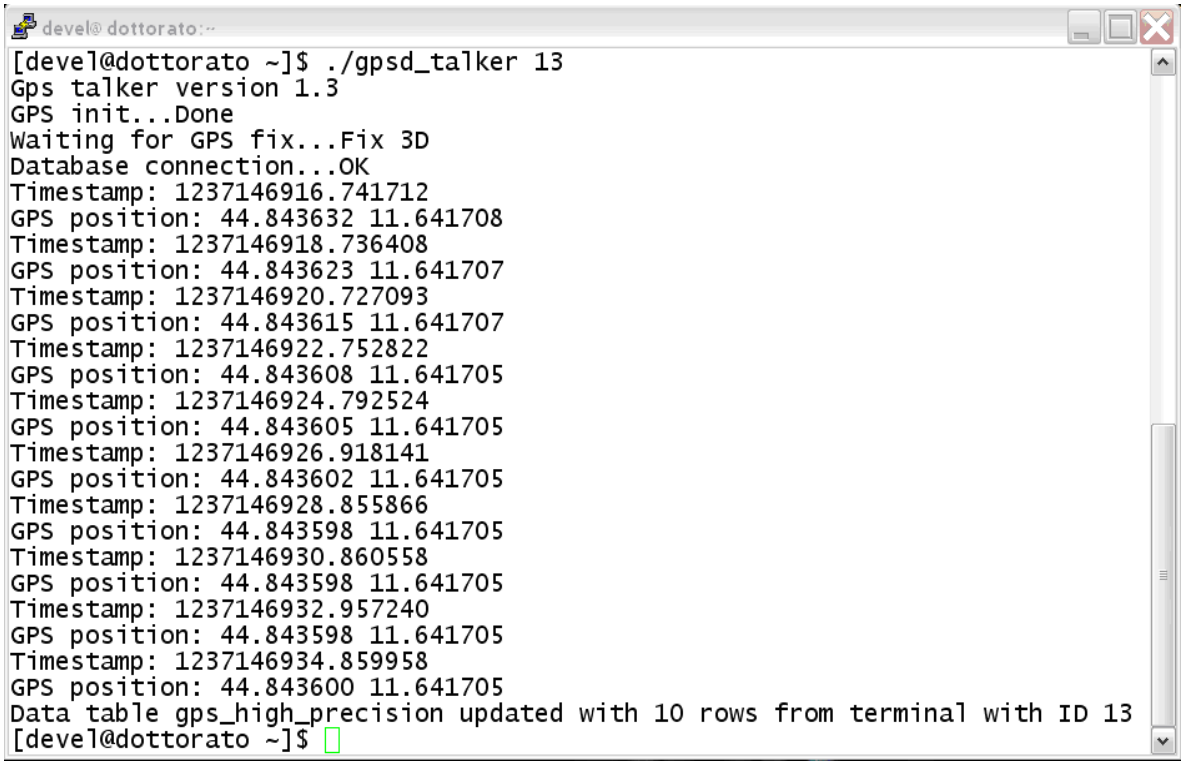


Figura 4.10: Terminale GPS: inserimento dati

Per quanto riguarda il caso con un parametro, visivamente è molto simile al precedente, la differenza sta nel fatto che in tal caso è stata anche inizializzata la connessione al database MI System via rete, e sono stati scritti dieci dati GPS di posizione ricevuti nella tabella creata dinamicamente in precedenza.

In Figura 4.11 è mostrato lo screenshot relativo alla chiamata con due parametri:

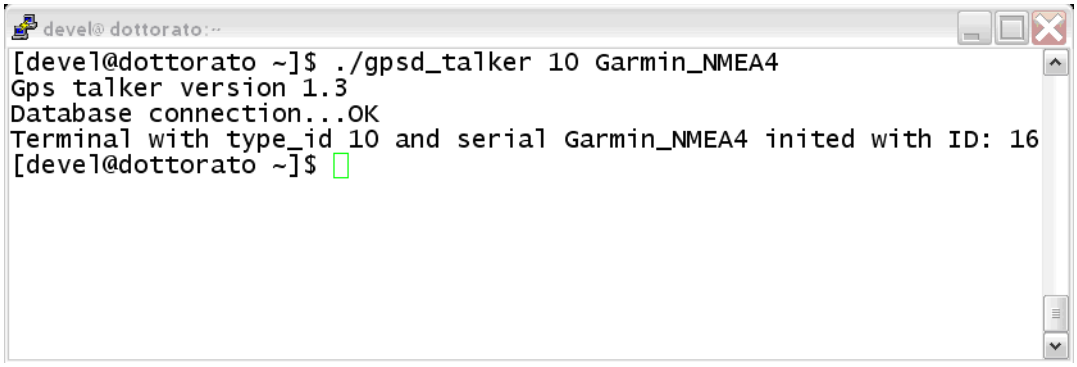


Figura 4.11: Terminale GPS: inizializzazione nuovo terminale

In questo caso il software si collega al database MI System e richiede la attivazione di una nuova istanza di un tipo di terminale in precedenza creato sempre dinamicamente. La nuova istanza di terminale viene inizializzata e le viene associato il nome o seriale richiesto e attribuito il primo identificativo di istanza disponibile del tipo richiesto che viene poi ritornato.

Infine in Figura 4.12 è presente lo screenshot del caso di utilizzo con tre parametri:

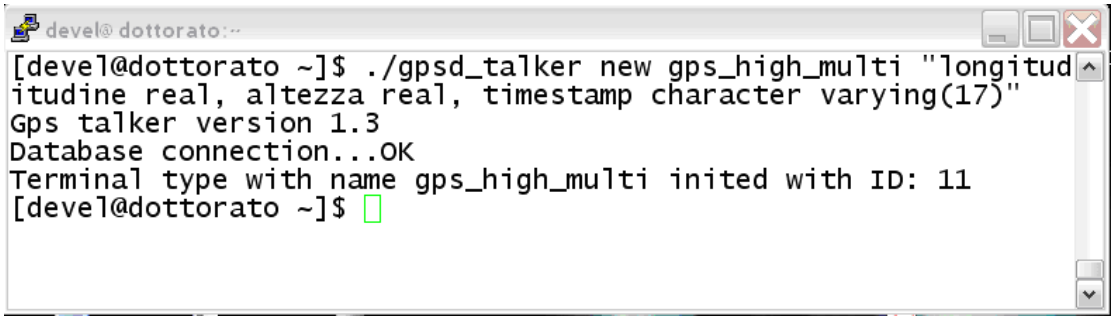


Figura 4.12: Terminale GPS: inizializzazione nuovo tipo di terminale

In quest'ultimo caso si vede come un nuovo tipo di terminale venga inizializzato all'interno del database assegnandogli il primo ID di tipo libero, ed associando a questo dinamicamente una struttura dati adatta a contenere il tipo di dati che verranno poi inviati.

4.12 Task 0 - Conclusioni

Anche se non è stato completamente implementato il sistema di gestione flotte progettato, i cui schemi sono in Appendice A, ovvero il Mobile Information System, la parte realizzata come Proof of Concept rappresenta un vero e proprio sistema completo di database su RDBMS e

software di gestione in grado di interagire con uno (o più) terminali GPS memorizzandone i dati.

La parte realizzata dimostra compiutamente le possibilità descritte nel progetto iniziale, avendo implementato tutti i concetti cardine in un prototipo funzionante.

Le finalità del Task 0 sono dunque pienamente raggiunte.

È stato infatti realizzato un sistema di storicizzazione di dati da terminali remoti con gestione di database e con relativa parte applicativa, entrambe correttamente funzionanti, e che possono essere prese a riferimento per sviluppi futuri per formulare una proposta compiuta di Standard Internazionale.

4.13 Task 0 - Sviluppi futuri

Oltre la Proof of Concept oggetto del presente Dottorato, si possono immaginare alcuni passi successivi che porterebbero il Mobile Information System a compimento. Ad esempio:

- Aggiunta di interazione con terminali diversi e tramite differenti reti, ad esempio GPRS, UMTS per sistemi geograficamente dislocati, oppure Wi-Fi/Max
- Sviluppo di un software di gestione modulare che consenta operazioni complesse oltre alla semplice storicizzazione dei dati
- Sviluppo di interfacce grafiche per una interazione semplificata con il Mobile Information System
- Aggiunta di gestione dinamica anche per i protocolli di comunicazione nel database e negli applicativi per maggiore flessibilità
- Aggiunta di funzionalità di sicurezza a tutta la infrastruttura informatica
- Definizione della struttura del database come Standard Internazionale (ISO)

Capitolo 5

5.1 Task 1

Il Task 1 ha riguardato lo studio del problema di realizzare una trasmissione sicura di dati attraverso un network ISOBUS ed in seguito attraverso un gateway dedicato e Internet per comunicazioni long range per diagnosi e fleet management. Attualmente tutte le comunicazioni che avvengono a bordo di una macchina agricola tramite bus ISOBUS sono in chiaro. Inoltre la modalità di funzionamento del bus stesso rende semplice la intercettazione di tutte le informazioni circolanti. Ad esempio è possibile che una centralina collegata al bus rilevi messaggi dedicati ad altri componenti collegati. Oltre a ciò la sicurezza e riservatezza dei dati deve essere mantenuta anche nella parte di storicizzazione ed eventualmente gestione dei dati stessi.

In Figura 5.1 è presente uno schema che permette di analizzare più in dettaglio il tipo di sistema che si vorrebbe realizzare:

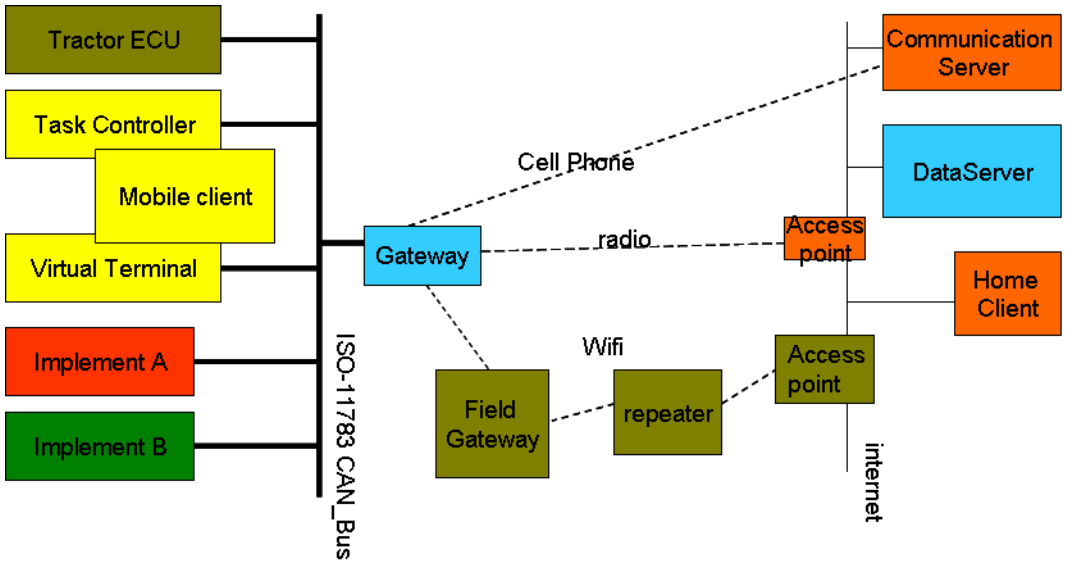


Figura 5.1: Comunicazioni long range in ambito agricolo

In figura a sinistra è rappresentata la rete in standard ISOBUS a cui sono collegati vari dispositivi, ad esempio Tractor ECU, Task Controller, Virtual Terminal e vari Implement, oltre ad un Mobile Client. Questi dispositivi tipicamente sono prodotti da costruttori diversi. Tramite un gateway dedicato collegato ad Internet e varie tecnologie di comunicazione long range (ponte radio dedicato, Wi-Fi, tecnologie GSM e GPRS) si vorrebbe che dati prodotti da uno o più dispositivi collegati ad ISOBUS raggiungessero in maniera sicura un sistema di

gestione remoto rappresentato a destra in figura, quale ad esempio quello progettato nel Task 0, e viceversa. Si vorrebbe inoltre che i dati prodotti venissero anche gestiti in maniera sicura, consentendo che la conservazione e elaborazione dei dati stessi avvenisse anche nella medesima locazione remota senza che costruttori o utenti finali diversi potessero ottenere accesso a informazioni di proprietà di altri soggetti. Tutto ciò è attualmente impossibile, visto che i dati viaggiano in chiaro, e dunque chiunque abbia la possibilità di accedervi durante una qualsiasi delle fasi legate al loro trasporto, elaborazione e conservazione è in grado di ottenere l'accesso alle informazioni.

5.2 Task 1 - Alcune considerazioni e problemi

Per mantenere la sicurezza e la riservatezza dei dati è necessario che tutta la catena di trasmissione e memorizzazione sia resa sicura. Per semplicità di ragionamento procederemo considerando le varie fasi a partire dalla generazione dei dati stessi fino alla loro 'consegna' al sistema di gestione remoto.

Partendo dalla sorgente la prima immediata considerazione è che vista la struttura della rete interna nelle macchine agricole la crittografia è indispensabile a livello di 'generazione' dei dati allo scopo di proteggere le informazioni trasportate nel segmento di rete ISOBUS. Infatti per mantenere la riservatezza dei dati, visto che qualsiasi dispositivo può leggere quanto circola sul bus, e molto probabilmente i vari dispositivi sono costruiti da produttori diversi e possono anche essere gestiti da soggetti diversi (si pensi ad esempio al noleggio di accessori per trattori), è necessario che ogni centralina o dispositivo collegati al bus crittografi i dati prima di inviarli sul bus stesso. Ciò ovviamente per i soli dati per i quali ha senso farlo, non certamente per informazioni real-time dedicate necessarie al funzionamento della macchina agricola.

Per questo motivo è necessario che gli algoritmi di cifratura siano molto efficienti, può essere necessario crittografare anche interi file su hardware embedded quale quello utilizzato tipicamente per centraline ISOBUS, che tipicamente non ha elevate prestazioni in termini di potenza di elaborazione. I sistemi elettronici destinati a mezzi mobili devono infatti tener in particolare conto la questione dei consumi, e ciò ne limita le prestazioni ottenibili in termini di potenza di calcolo.

Altro problema è che attualmente ISOBUS non prevede crittografia a livello di protocollo, e quindi senza modifiche non è possibile implementare funzionalità per rendere sicura la

comunicazione. E' anche vero che una modifica al protocollo di base sarebbe molto pesante da implementare, dovendo prevedere con tutta probabilità la modifica per compatibilità di tutta la base dei dispositivi attualmente in uso. E' ovvio che tale soluzione comporta notevolissimi problemi. La scelta di cifrare i dati da trasmettere all'origine, ovvero all'interno delle singole centraline collegate ad ISOBUS però risolverebbe in un colpo solo gran parte dei problemi. Infatti i dati così protetti non solo sarebbero in grado di viaggiare in maniera sicura all'interno della macchina agricola, ma lungo tutto il percorso tramite Internet e fino a giungere al sistema di gestione remota. Nel sistema di gestione remota poi potrebbero essere immagazzinati in maniera sicura o decifrati opportunamente in base all'utilizzo e alla struttura stessa del sistema di gestione. Ad esempio si potrebbe pensare di separare le parti di database in modo che siano accessibili solo ai soggetti aventi diritto. In ogni caso la cifratura all'origine rimane l'unica possibilità per rendere effettivamente sicure le comunicazioni. Rimane aperto ancora il problema del trasporto dei dati, in quanto ISOBUS prevede la possibilità di comunicare esclusivamente con centraline connesse al suo bus, e non con dispositivi remoti. E' quindi necessario anche progettare una estensione del protocollo che supporti comunicazioni con host remoti collegati su rete TCP/IP, in modo da rendere possibile la trasmissione dati via Internet e la memorizzazione degli stessi in un sistema di gestione remoto basato su strumenti hardware e software standard e quindi con alte prestazioni e basso costo. Nel prossimo paragrafo verranno introdotti alcuni cenni alla moderna crittografia, per poi passare alla soluzione ideata ed oggetto di proposta.

5.3 Cenni di crittografia moderna

Allo scopo di fornire alcuni strumenti utili per chiarire la soluzione proposta per il Task 1 vengono nei seguenti paragrafi presentati alcuni cenni di crittografia.

Partiamo da alcune definizioni generali per poi esplorare alcuni algoritmi di crittografia comunemente utilizzati.

Se M è il messaggio da cifrare, C il messaggio cifrato, K la chiave, E la 'funzione' di cifratura e D quella di decifratura, e A e B sono due soggetti che intendono scambiarsi informazioni, nella crittografia moderna ci sono essenzialmente due tipi di cifratura, simmetrica ed asimmetrica.

5.3.1 Crittografia Simmetrica

In Figura 5.2 è schematizzata la cifratura simmetrica.

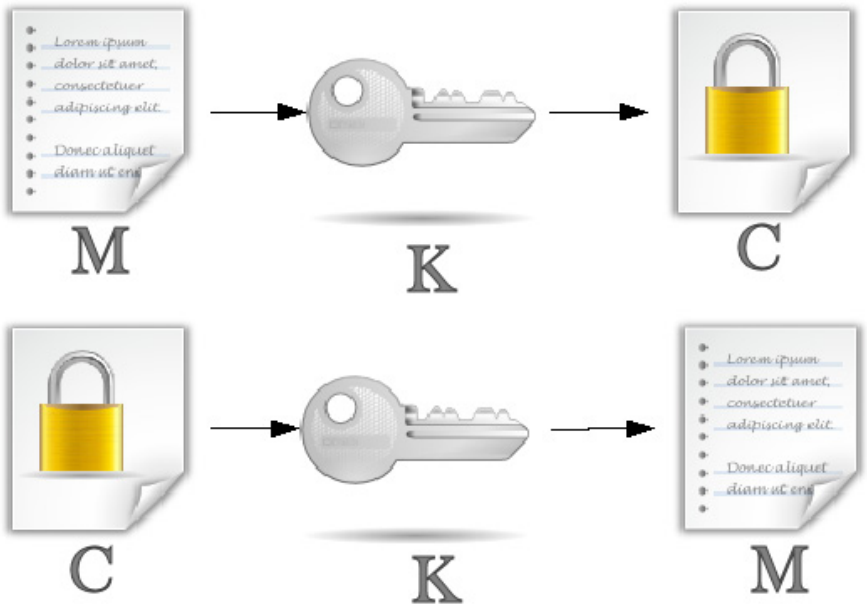


Figura 5.2: Cifratura simmetrica

Nella Crittografia Simmetrica esiste una unica chiave utilizzata per cifrare e decifrare il messaggio. C = E (M, K) and M = D (C, K). La chiave K deve essere tenuta segreta, quindi deve essere concordata in anticipo o trasmessa su di un canale sicuro. Questa modalità è comunemente utilizzata per scambi sicuri di informazioni, ma presenta le seguenti controindicazioni:

- Le chiavi devono essere comunicate in segreto, attraverso un canale sicuro
- Se la chiave è compromessa (ovvero un attaccante riesce a entrarne in possesso) allora tutti messaggi possono essere compromessi, ovvero decifrati e/o sostituiti
- Se si devono comunicare messaggi cifrati a molti utenti, è necessario concordare una chiave per ogni coppia di utenti, in modo che un destinatario non possa decifrare messaggi inviati ad un altro destinatario. Tuttavia, il numero delle chiavi da utilizzare e distribuire diventa velocemente molto grande.

Nonostante i problemi evidenziati la cifratura simmetrica è spesso utilizzata in quanto richiede minore potenza computazionale rispetto ad altre tecniche.

Alcuni algoritmi di crittografia simmetrica sono: Data Encryption Standard (DES) e varianti (3DES ad esempio), Advanced Encryption Standard (AES). In bibliografia sono presenti alcuni riferimenti per approfondire la conoscenza di tali algoritmi.

5.3.2 Crittografia Asimmetrica

Per superare alcuni limiti della cifratura simmetrica è possibile utilizzare una cifratura detta per differenza asimmetrica, esemplificata nella seguente Figura 5.3:

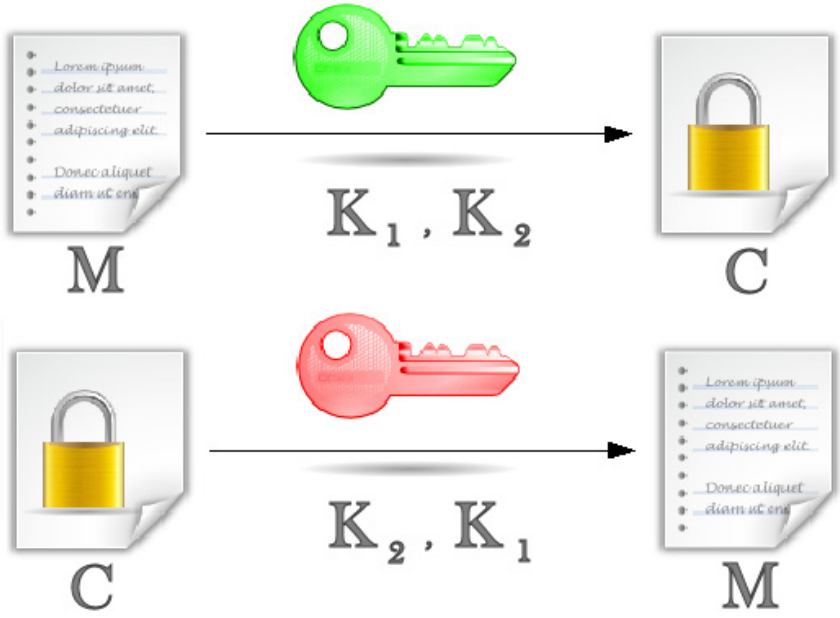


Figura 5.3: Cifratura asimmetrica

Nella Crittografia Asimmetrica ci sono due chiavi, K1 e K2, dette pubblica e privata, che consentono di ottenere vari risultati. La chiave pubblica è nota a chiunque, la privata solo al suo proprietario. Si ha che C = E (M, K1) ed M = D (C, K2). A cifra M trasformandolo in C, e lo invia a B. Se K1 è la chiave pubblica e K2 è la privata, e A è il soggetto sorgente che cripta il messaggio M in C con la chiave pubblica K1 del destinatario B si ottiene che solo B possa leggere l'informazione decifrandola con la sua chiave privata K2 (ciò consenta ad esempio di scambiarsi una chiave segreta e procedere con crittografia simmetrica). Nel caso invece che K1 sia la chiave privata e K2 la pubblica, A ottiene il risultato di 'firmare' il messaggio, in quanto questo potrà essere decodificato con la sua sola chiave pubblica (tipico esempio di firma digitale).

Si supponga che un attaccante possa intercettare tutte le comunicazioni tra A e B. Egli ad esempio intercetta la chiave pubblica e il messaggio cifrato durante le fasi intercorse nel primo

esempio descritto. Tuttavia non può decifrare il messaggio, perché è necessaria la chiave privata, che non viene mai trasmessa. Inoltre, diversi utenti possono utilizzare la stessa chiave pubblica di B per cifrare messaggi, perché la conoscenza della chiave pubblica non permette la decifrazione. Con questo sistema si risolvono di colpo i due problemi della crittografia simmetrica relativi alle chiavi da distribuire; in pratica:

- Le chiavi pubbliche possono essere trasmesse in un modo qualsiasi, la sicurezza delle trasmissione non dipende da queste, anzi devono essere note a chiunque voglia comunicare con il proprietario della chiave. Esistono varie tecnologie a supporto dello scambio di chiavi che può essere anche automatizzato.
- Se la chiave pubblica viene intercettata non c'è nessun problema, anzi è il normale funzionamento della cifratura asimmetrica. La chiave privata invece deve rimanere assolutamente riservata ma non bisogno di trasmetterla. Per un attaccante in pratica è impossibile intercettare l'unica chiave utile per la decifratura lungo il canale di trasmissione.
- Se si devono comunicare messaggi cifrati a molti utenti, non è necessario concordare una chiave segreta su canale sicuro per ogni coppia di utenti, ma basta rendere pubbliche le rispettive chiavi.

Esistono vari tipi di algoritmi per la crittografia asimmetrica che si basano su differenti principi matematici. Ad esempio RSA (da Rivest, Shamir and Adleman, primi autori a descrivere pubblicamente l'algoritmo) e Rabin (dal nome del suo autore) si basano sulla fattorizzazione di numeri interi, ovvero sulla difficoltà di fattorizzare un numero formato dal prodotto di due numeri primi sufficientemente grandi. Invece ElGamal (dal nome del suo autore) e DHIES (Diffie-Hellman Integrated Encryption Scheme) si basano su di un problema matematico detto 'problema del logaritmo discreto', in inglese Discrete Logarithm Problem (DLP). Il problema maggiore della cifratura asimmetrica è che la potenza computazionale richiesta è molto maggiore che nel caso della simmetrica. In bibliografia sono presenti riferimenti che consentono di approfondire gli algoritmi citati.

5.3.3 Algoritmi ibridi

Sembrerebbe che la crittografia asimmetrica, sia molto migliore di quella simmetrica, in virtù dei vantaggi illustrati precedentemente. Tuttavia non è esente da alcuni problemi come la lentezza: gli algoritmi simmetrici sono molto più veloci di quelli asimmetrici nel cifrare e decifrare i dati, che di fatto sono piuttosto 'pesanti' dal punto di vista computazionale.

Per ovviare a questo problema si utilizzano algoritmi ibridi che combinano i vantaggi degli algoritmi simmetrici e di quelli asimmetrici. Questa può essere una buona soluzione anche per il campo agricolo, e nel seguito sarà discusso il motivo per cui è molto importante riuscire ad evitare di utilizzare la crittografia simmetrica.

L'idea consiste nel generare una chiave simmetrica casuale e cifrarla con la chiave pubblica asimmetrica del destinatario. Il mittente invierà la chiave cifrata. Il destinatario decifrerà con la propria chiave privata asimmetrica la chiave simmetrica casuale. La procedura potrebbe essere la seguente:

1. B invia ad A la sua chiave pubblica
2. A genera una chiave simmetrica casuale K e la cifra con la chiave pubblica di B
3. B riceve la chiave K cifrata e la recupera con la sua chiave privata
4. A questo punto sia B che A si sono accordati su una comune chiave K simmetrica con cui cifrare i messaggi

I vantaggi sono diversi: per prima cosa le comunicazioni saranno cifrate e decifrate con la chiave simmetrica K, molto più veloce e lo scambio della chiave K avviene in modo sicuro; solo il destinatario potrà decifrarla. Inoltre, per una maggiore sicurezza, si può generare una nuova chiave K quando si desidera (ad esempio al trascorrere di un certo intervallo o in funzione di una certa quantità di dati scambiati) e la si può inviare con lo stesso sistema. La chiave K viene generalmente detta chiave di sessione.

5.3.4 Crittografia in ISOBUS - ulteriori problemi

Riassumendo l'utilizzo di crittografia in un sistema in standard ISOBUS presenta svariate criticità. Sicuramente uno dei problemi è legato alla limitata potenza di calcolo disponibile, che limita l'utilizzo della crittografia asimmetrica nelle attuali centraline. Oltre a ciò analizzando il funzionamento dei vari tipi di crittografia emerge anche una altro problema. Se fossimo effettivamente costretti ad utilizzare esclusivamente crittografia simmetrica sarebbe necessario cablare nel software delle centraline le chiavi necessarie alla cifratura. E' evidente che tale approccio presenta una notevole criticità. E' evidente che non sarebbe saggio utilizzare una chiave unica, ad esempio per tutte le centraline di un singolo produttore perché

la compromissione di questa singola chiave comporterebbe la perdita di segretezza su tutti i dati prodotti sino a quel momento e la necessità di dover modificare la chiave in tutte le centraline ed i sistemi già funzionanti, una ipotesi molto difficilmente realizzabile. Immaginare di utilizzare chiavi diverse per il software di ogni singola centralina comporterebbe notevoli difficoltà per la gestione e la programmazione delle chiavi, tanto da doversi ritenere anch'essa una strada difficilmente praticabile.

5.4 ISOBUS - Comunicazione long range e gestione dati sicure - la soluzione

Per comodità elenchiamo di nuovo riassumendoli in punti i problemi emersi sinora e che impediscono di ottenere il risultato desiderato, ovvero trasmettere e gestire remotamente dati generati su rete ISOBUS in maniera sicura.

1. ISOBUS non è dotato nativamente di funzionalità per comunicazione sicura
2. ISOBUS non è dotato nativamente di funzionalità per la comunicazione con sistemi remoti in particolare su rete standard TCP/IP
3. E' necessario trasferire dati generati in reti ISOBUS in quantità anche rilevanti fino ad un gateway e poi tramite Internet fino ad un sistema di gestione remoto
4. Può essere necessario che i dati da trasferire siano crittografati 'on the fly'
5. Per implementare efficacemente funzionalità crittografiche, ed in particolare per la gestione delle relative chiavi sarebbe utile poter utilizzare algoritmi con crittografia asimmetrica ma la attuali centraline ISOBUS non sono dotate di sufficiente potenza di calcolo

Vediamo quindi come tutte le criticità elencate vengano risolte dalla proposta elaborata nel Task 1.

5.4.1 I punti 1, 2 e 3

Per quanto riguarda i problemi introdotti e citati ai punti 1, 2 e 3 del precedente paragrafo risulta che senza dubbio sono necessarie delle modifiche ad ISOBUS per consentire l'implementazione di nuove funzionalità quali quelle descritte nel corso del presente capitolo, che non erano state previste nella definizione originale dello Standard. E' di tutta evidenza che l'implementazione di tali modifiche potrebbe essere estremamente onerosa se si dovesse applicare a tutto il parco macchine esistente. Per questo motivo è assolutamente da ricercarsi

una soluzione alternativa che sia in grado di mantenere la compatibilità con il passato consentendo però di ottenere le nuove funzionalità richieste. Per ottenere questo risultato una via esiste, e consiste nell'utilizzare una caratteristica già presente nello Standard brevemente descritta nel seguente paragrafo, apportando modifiche che risultino trasparenti ai sistemi già in produzione.

5.4.1.1 I protocolli di trasporto in ISOBUS

ISOBUS nel suo ampio parco di funzionalità è dotato anche di un sotto-protocollo di trasporto detto Transport Protocol (TP), descritto nella norma ISO 11783 part 3, ovvero "Tractors and machinery for agriculture and forestry -- Serial control and communications data network -- Part 3: Data link layer".
Esiste anche una estensione al TP detta Extended Transport Protocol (ETP), descritta nella norma ISO 11783 part 6, ovvero "Tractors and machinery for agriculture and forestry -- Serial control and communications data network -- Part 6: Virtual terminal", Annex K.
I documenti appena citati non sono liberamente riproducibili, ed anzi hanno un costo decisamente elevato. Nel seguito verranno presentati quindi solo alcuni cenni al funzionamento dei protocolli in modo da consentire la comprensione della proposta oggetto del Task 1. In bibliografia alla sezione Internet sono presenti riferimenti per l'acquisto ed il download dei documenti citati che consentono di approfondire il tema. E' utile ricordare che la normativa è attualmente ancora in fase di sviluppo per cui alcuni documenti potrebbero essere in forma non definitiva.

5.4.1.2 ISOBUS Transport Protocol

Le funzioni principali del Transport Protocol sono due:

1. Pacchettizzazione e riassemblaggio dei messaggi
2. Gestione delle connessioni

Per quanto riguarda il punto uno bisogna dire che messaggi con dimensione maggiore di otto bytes non sono allocabili in un singolo Data Frame CAN normalmente utilizzato per le comunicazioni ISOBUS. Dunque messaggi lunghi devono essere spezzati in pacchetti più piccoli, ed ogni pacchetto deve essere spedito in Frame diversi. In fase di ricezione poi ogni

singolo Frame deve essere ricevuto e processato in modo da riassemblare il pacchetto originario. Tale comportamento è tipico delle reti a pacchetti, ed è ad esempio simile a quanto accade per il protocollo TCP normalmente utilizzato per connessioni via Internet.

Il Data Frame CAN include un campo dati lungo otto bytes. Siccome ogni pacchetto in cui viene suddiviso un messaggio di grandi dimensioni deve essere individualmente identificato, in modo da poter correttamente riassemblare il messaggio stesso, il primo byte del campo dati viene utilizzato come numero di sequenza (sequence number) del pacchetto. Per tale motivo ad ogni pacchetto viene assegnato un numero di sequenza che può variare tra 0 e 255. Dunque la massima dimensione di un messaggio diventa (255 pacchetti x 7 bytes/pacchetto) = 1785 bytes.

Per quanto riguarda il riassemblamento dei pacchetti essi verranno ricevuti in sequenza date le caratteristiche di una rete ISOBUS. Quindi ogni pacchetto facente parte di un singolo messaggio dovrà essere assemblato in ordine di sequence number, in una singola sequenza di bytes. Al termine del procedimento questa sequenza di bytes sarà passata alla applicazione che ha richiesto il messaggio di grandi dimensioni. E' importante notare che contrariamente a quanto accade per altri tipi di dati in questo caso non c'è nessuna garanzia sulle tempistiche di consegna, come vedremo infatti la gestione delle connessioni prevede anche il rifiuto o l'interruzione di una comunicazione. Ciò va tenuto in conto ma non vanifica l'utilizzo del transport protocol per i nostri scopi, dai quali sono escluse informazioni real-time, ma si pensa di trasmettere dati per diagnosi e fleet management che non hanno tipicamente vincoli stringenti in fatto di tempistiche di consegna, anche considerando che fuori dalla rete ISOBUS gli stessi dati dovranno passare attraverso Internet con ritardi ordini di grandezza superiori a quelli tipici di ISOBUS. In ogni caso nel seguito sono presenti alcuni cenni sul punto due, ovvero la gestione delle connessioni, al fine di meglio comprendere l'indeterminatezza associata ai tempi di trasmissione.

Nel seguito si farà riferimento con il termine 'sender' al controller ISOBUS che trasmette un messaggio di 'Request to Send' (nel seguito RTS) ovvero richiesta di invio, mentre 'receiver' sarà il controller che in risposta invierà un messaggio 'Clear to Send' (nel seguito CTS) ovvero invio accettato.

La gestione delle connessioni (Connection Management) riguarda l'aprire, utilizzare e chiudere delle connessioni 'virtuali' per trasferimenti tra specifici controller. Una connessione virtuale in ambiente ISO 11783 si può considerare come una associazione temporanea tra due controller allo scopo di trasferire un singolo messaggio di grandi dimensioni. Nel caso invece

la connessione sia da uno a molti (broadcast) non è previsto controllo di flusso né è prevista una chiusura esplicita della connessione virtuale stessa.
Una connessione si considera iniziata nel momento in cui un controller trasmette un messaggio RTS ad un indirizzo di destinazione. Il messaggio RTS contiene la dimensione dell'intero messaggio in bytes, il numero di pacchetti complessivo tramite i quali il messaggio verrà trasferito, il massimo numero di pacchetti che possono essere spediti in risposta ad un singolo CTS, ed un'altro parametro ISOBUS, ovvero il Parameter Group Number del messaggio trasportato. Al momento della ricezione di un messaggio RTS, un controller può decidere di accettare la connessione o rifiutarla. Per accettare la connessione il receiver trasmetterà un messaggio CTS. Il CTS inviato contiene il numero di pacchetti del messaggio che il receiver può accettare, ed il sequence number del primo pacchetto atteso. Il receiver deve assicurarsi di aver disponibili sufficienti risorse per gestire il numero di pacchettti dei quali sta accettando la consegna. Il sequence number del pacchetto nel caso di una connessione appena aperta, sarà uno.
E' da notare che il CTS potrebbe non contenere l'accettazione di tutti i pacchetti che compongono il messaggio completo.
Per rifiutare la connessione un controller deve rispondere con un messaggio di Connection Abort (termine connessione). La connessione può essere rifiutata per una ragione qualsiasi, anche se tipicamente mancanza di risorse, ad esempio memoria ne è la causa.
La connessione si considera stabilita per il sender quando questi riceve un CTS corrispondente al suo RTS dal receiver. La connessione si considera stabilita per il receiver nel momento in cui questi ha trasmesso con successo il suo CTS in risposta al RTS ricevuto.
Queste definizioni sono utili a determinare quando un Connection Abort deve essere inviato per chiudere una connessione. Un receiver deve mandare un Connection Abort se ha letto il messaggio RTS e decide di non stabilire la connessione. Ciò permette di evitare che il sender debba attendere un timeout prima di cercare di attivare una nuova connessione.
Vediamo ora qualcosa sul trasferimento dati, che inizia dopo che il sender ha ricevuto il messaggio CTS. Una eccezione è nel caso in cui il trasferimento dati avviene in seguito ad un messaggio del tipo 'Broadcast Announce' (annuncio di trasmissione broadcast), in questo caso il messaggio CTS non è usato. Nel caso invece di messaggi tra due soli controller, il receiver è responsabile della gestione del flusso tra i due controller coinvolti nella trasmissione. Se il receiver vuol fermare temporaneamente il flusso dati, mantenendo la connessione aperta, deve utilizzare un CTS settando il numero di pacchetti che è disponibile a ricevere eguale a zero.

Nel caso il flusso debba essere fermato per alcuni secondi il receiver deve ripetere la trasmissione del CTS ogni mezzo secondo per avvisare il sender che la connessione non si è interrotta.

Vediamo ora qualcosa sulla chiusura delle connessioni. Esistono due casi di chiusura in assenza di errori. Il primo occorre nel caso di trasmissione broadcast, il secondo nel caso di connessione punto a punto. Nel primo caso non viene effettuata nessuna chiusura esplicita della connessione, che viene considerata chiusa in seguito alla semplice ricezione dei dati previsti. Nel caso di connessione punto a punto invece ed al momento della ricezione dell'ultimo pacchetto previsto, il receiver trasmette un 'end of message ack' (avviso di fine messaggio per ricevuta) al sender. Ciò per segnalargli che la connessione viene considerata chiusa dal receiver. Il 'end of message ack' serve a far sì che la connessione venga liberata per successivi utilizzi da parte di altri controller.

Nel caso di trasmissione broadcast i receiver non devono inviare un messaggio di Connection Abort. Invece nel caso di connessione punto a punto sia il sender che il receiver possono, in qualsiasi momento, utilizzare un Connection Abort per terminare la connessione.

Ad esempio se il receiver dovesse accorgersi che non sono rimaste risorse sufficienti per processare il messaggio stesso, può semplicemente terminare la connessione emettendo un messaggio di Connection Abort. Al ricevimento del Connection Abort, tutti i pacchetti costituenti il messaggio di grandi dimensioni verranno scartati.

Altra causa di chiusura di una connessione per timeout può essere il guasto di uno dei controller coinvolti nel caso di trasmissione punto a punto. Quando il sender o il receiver decidono di chiudere una connessione per qualsiasi ragione incluso il timeout, devono spedire un messaggio di Connection Abort. A titolo di esempio nella seguente Tabella 5.4 sono elencati alcuni motivi per cui viene effettuato un Connection Abort con i codici associati secondo lo Standard ISOBUS:

Value	Description
1	Already in one or more connection managed sessions and cannot support another.
2	System resources were needed for another task so this connection managed session was terminated.
3	A timeout occurred and this is the connection abort to close the session.
4–250	Reserved for assignment in a future International Standard.
251–255	Per ISO 11783 Parts 7 & 8 definitions.

Tabella 5.4: ISOBUS - Motivi e codici di Connection Abort

Lo Standard ISOBUS definisce in ogni dettaglio come devono avvenire le transazioni associate al Transport Protocol, in questo paragrafo si è semplicemente voluto evidenziare alcune caratteristiche fondamentali, ovvero che è un protocollo pensato per il trasporto di dati punto a punto o anche broadcast, che la dimensione massima di un singolo messaggio trasportabile è di 1785 bytes, e che per quanto lo Standard cerchi di minimizzare tempi di attesa nella consegna dei dati, le tempistiche non sono assolutamente garantite, anzi in qualche modo il TP è da considerare a più bassa priorità rispetto al normale traffico ISOBUS, visto che può essere interrotto per qualsiasi ragione in qualsiasi momento. Nel seguente paragrafo verranno introdotti alcuni cenni ad una estensione del TP, detta Extended Transport Protocol, visto che può essere utile alla soluzione dei problemi analizzati nel Task 1.

5.4.1.3 ISOBUS Extended Transport Protocol

Le funzioni di Extended Transport Protocol (ETP) sono richieste per trasferire messaggi di lunghezza maggiore di quelle specificata da TP, ovvero 1785 bytes. Per dimensioni inferiori lo Standard ISOBUS prevede che venga utilizzato il TP definito in ISO 11783-3.
ETP usa quattro byte di controllo addizionali tra quelli presenti nella lista riservata alla gestione delle connessioni nel TP per i messaggi di richiesta e risposta di connessioni ‘estese’, ovvero secondo lo standard ETP.
I byte di controllo utilizzati sono:
- Byte di controllo 20 - messaggio ETP ‘Request To Send’
- Byte di controllo 21 - messaggio ETP ‘Clear To Send’
- Byte di controllo 22 - messaggio ETP ‘Data Packet Offset’
- Byte di controllo 23 - messaggio ETP ‘End of Message Acknowledgement’

Invece il normale messaggio di Connection Abort viene usato anche per ETP, mentre i messaggi TP di trasmissione broadcast non devono essere usati per ETP.
Il massimo numero di byte trasmissibili tramite una singola connessione ETP è di 117440512 (2 ^ 24 * 7) bytes, ovvero 112Mb.
Altra caratteristica interessante è che una sessione ETP può essere aperta in parallelo con una sessione TP. Il sender che attiva in parallelo sessioni TP ed ETP deve aspettarsi che il receiver possa mettere in pausa una o entrambe le sessioni per un tempo arbitrario, e quindi non è neppure possibile stabilire a priori quale delle sessioni terminerà per prima. Il comportamento rispetto a priorità e ritardi di ETP è identico a quello di TP.

L'utilità di ETP risalta immediatamente pensando al File Transfer Protocol (FTP) compatibile con la suite standard TCP/IP. Per analogia con tale protocollo si può immaginare di implementare una sessione di controllo su TP, mentre per il canale per il trasporto dei dati veri e propri si può utilizzare ETP, i grado di trasferire anche rilevanti quantità di informazioni con una singola sessione.

5.4.2 Soluzione per i punti 1, 2 e 3

Per risolvere le questioni ai punti 1,2 e 3 è possibile immaginare di estendere il protocollo ISOBUS in modo da incapsulare nei messaggi del Transport Protocol ed Extended Transport protocol dei nuovi data frame concepiti in modo da assolvere alle funzioni richieste. In questo modo i sistemi attualmente in produzione non verrebbero toccati dalle nuove modifiche mantenendo piena compatibilità, ma sarebbe possibile introdurre le nuove funzionalità richieste semplicemente aggiungendo alle reti ISOBUS esistenti le nuove centraline. Un tipico esempio potrebbe essere l'aggiunta di un gateway sul bus in grado di trasferire dati tramite Internet ad un sistema di gestione remota.
Le estensioni da aggiungere incapsulandole in TP/ETP dovrebbero essere ad esempio del tipo:

1. Supporto dell'indirizzamento TCP/IP (Internet), che comprenda modalità Standard per trasmettere gli indirizzi IP verso i quali si vuole aprire una connessione al gateway. Il gateway diventa la macchina che implementando le estensioni allo standard ISOBUS gestisce effettivamente la comunicazione con reti TCP/IP. Tale comunicazione può essere effettuata tramite uno qualsiasi dei canali disponibili, cablati o wireless.
2. Estensioni che prevedano la possibilità di richiedere connessioni di vario tipo tra quelle offerte dalla suite TCP/IP.
3. Un protocollo incapsulato che utilizzi una connessione virtuale TP per controllo ed una connessione virtuale ETP concorrente per trasporto dati del tipo di File Transfer Protocol (FTP) compatibile con suite TCP/IP.
4. Per quest'ultimo protocollo di trasferimento dati una funzionalità del tipo del resume presente nelle connessioni FTP. In tal modo nel caso di Connection Abort sopravvenuto nel mezzo di una connessione virtuale ETP i dati scambiati sino a quel momento potrebbero essere conservati dal gateway in modo da non doverli ritrasmettere in toto. Molto utile nel

caso di trasferimenti di file di grosse dimensioni che potrebbero richiedere tempistiche lunghe per la trasmissione e quindi essere facilmente interrotti.

In tutti i casi accennati la cosa fondamentale è che incapsulando le estensioni nei messaggi relativi a TP/ETP si ottiene il risultato di mantenere la compatibilità con il passato e contemporaneamente di riuscire ad implementare le nuove funzionalità richieste. I vincoli di TP ed ETP in termini di tempistiche e massima quantità di dati trasportati non sono rilevanti ai fini del risultato che si vuole ottenere, ovvero una trasmissione di dati attraverso un network ISOBUS ed in seguito attraverso un gateway dedicato e Internet per comunicazioni long range per diagnosi e fleet management. Per quanto riguarda la sicurezza della trasmissione essa verrà trattata nel successivo paragrafo.

5.4.3 Soluzione per i punti 4 e 5

Il problema di adottare crittografia in sistemi ISOBUS stante la disponibilità di tecnologie efficaci già esistenti è legato in sostanza all'aspetto delle prestazioni, oltre ad un discorso di estensione dello Standard ISOBUS attuale tramite l'aggiunta di nuove funzionalità da implementare sfruttando la stessa tecnica del caso precedente.
Per quanto riguarda l'aspetto delle prestazioni una soluzione possibile è intanto quella di utilizzare algoritmi di crittografia differenti a seconda delle operazioni da compiere.
Ad esempio si può immaginare per quanto riguarda la crittografia 'on the fly', qualora non fosse necessaria a livello di bus ISOBUS ma solo per proteggere le comunicazioni via Internet, di implementare un meccanismo di cifratura a livello del gateway, opportunamente dimensionato e quindi dotato di un hardware più performante dei normali controller. Ciò non altera significativamente il dimensionamento complessivo del sistema essendo il gateway stesso una singola centralina utilizzata in un nutrito insieme di altre. Il gateway potrebbe agire da interfaccia tra la macchina agricola e l'esterno implementando tecniche di cifratura complesse e senza particolari limitazioni, anche di tipo asimmetrico, volendo anche per conto di altri controller a bordo del veicolo, anche 'on the fly'.
Per quanto riguarda i rimanenti controller generici, sia nel caso di crittografia 'on the fly' che in tempo differito, una soluzione efficace sarebbe quella di utilizzare algoritmi ibridi, quali quelli descritti nel paragrafo 5.3.3. In questo modo si combinerebbe il vantaggio della crittografia simmetrica, ovvero la leggerezza computazionale, con quelli molto importanti nel

caso dell'ambiente ISOBUS, della crittografia asimmetrica, ovvero la maggiore semplicità di gestione delle chiavi. Facendo riferimento all'esempio riportato nel paragrafo 5.3.3 infatti se immaginiamo che A sia una centralina ISOBUS di un determinato produttore non c'è bisogno che tale centralina abbia cablata nel suo firmware nessuna chiave segreta, ma la sola chiave pubblica del costruttore, per inviare dati crittografati ad un ipotetico sistema di gestione. Infatti una volta generata una chiave segreta dinamicamente, essa potrebbe essere cifrata con la sola chiave pubblica spendendo una sola volta le risorse necessarie per crittografia asimmetrica, e spedita al sistema di gestione. In seguito la comunicazione potrebbe avvenire utilizzando la chiave segreta generata al costo computazionale della più leggera crittografia simmetrica. Per migliorare la sicurezza si può pensare di rigenerare la chiave segreta ogni tanto, ad esempio al trascorrere di un certo periodo di tempo, oppure in base alla quantità di dati spediti e crittografati con la chiave. Ancora si può pensare di utilizzare per la crittografia simmetrica vari tipi di algoritmi in base alle necessità di riservatezza dei dati da trasportare, scegliendo per le centraline meno performanti e/o i dati meno critici degli algoritmi computazionalmente più leggeri.
Adottando un approccio misto di questo genere il problema della crittografia oggetto del Task 1 sarebbe risolto.
Rimarrebbe aperta la questione della estensione di ISOBUS con la tecnica di incapsulamento delle modifiche in TP/ETP.
A titolo di esempio alcune estensioni utili sarebbero:

1. Estensioni che supportino la crittografia 'on the fly' operata dal gateway, per conto di altri controller collegati al bus o del gateway stesso.
2. Supporto per il trasporto di dati già crittografati all'interno dei controller su rete ISOBUS, in modo da consentire il trasferimento sicuro di dati criptati in origine.
3. Supporto per la selezione e l'utilizzo di differenti schemi ed algoritmi di cifratura in modo da poter realizzare il miglior compromesso tra costo computazionale e riservatezza.

5.5 Task 1 - Conclusioni

Le attività per il Task 1 svolte nel presente dottorato hanno compreso lo studio del complesso Standard ISOBUS attuale che prevede una comunicazione solo intraveicolare, che è stato messo poi concettualmente in relazione con lo scenario prospettato nel Task 0, ovvero il

progetto di un sistema ‘standard’ per fleet management in ambito agricolo. Ciò allo scopo di poter poi realizzare il progetto di massima di una soluzione che permetterebbe di implementare una comunicazione sicura nell’ambito del sistema complessivo costituito da un network ISOBUS e da un sistema di gestione dati remoto raggiunto via Internet. La soluzione prospettata ha numerosi e rilevanti vantaggi, tra cui:

- mantenimento della compatibilità con ambiente ISOBUS esistente grazie alla tecnica dell’incapsulamento in TP/ETP
- compatibilità con altri Standard molto diffusi come la suite TCP/IP
- tecnologie di crittografia già esistenti
- semplicità di gestione delle chiavi senza richiedere eccessiva potenza di calcolo grazie agli algoritmi di cifratura ibridi
- tecniche di crittografia selezionabili in modo da realizzare un compromesso tra potenza di calcolo richiesta e riservatezza dei dati
- crittografia ‘on the fly’ realizzata anche a livello di gateway, quindi maggiore flessibilità

Il lavoro svolto è attualmente in fase di valutazione. Durante il periodo di svolgimento del presente Dottorato è stata attribuita maggiore urgenza alle tematiche oggetto del successivo Task 2, su richiesta di ISO. Per questo motivo le attività riferite al Task 1 sono terminate dopo la realizzazione del progetto di massima, e si è proceduto nella elaborazione delle attività legate al Task 2 descritto nel prossimo capitolo.

5.6 Task 1 - Sviluppi futuri

Oltre il progetto di massima oggetto del presente Dottorato, si possono immaginare alcuni passi successivi che porterebbero il Task 1 a compimento. Tali passi sono stati già descritti nei paragrafi immediatamente precedenti, ovvero 5.4.2 punti da 1 a 4 e 5.4.3 punti da 1 a 3, pertanto non vengono qui ripetuti.

Capitolo 6

6.1 Task 2 - Background

Con il progresso della tecnologia e la diffusione di strumenti per la gestione delle informazioni direttamente a bordo dei veicoli in campo agricolo, la richiesta di equipaggiare le macchine con sistemi di gestione flotte (fleet management) e per la sincronizzazione tra veicoli ed accessori e tra differenti veicoli sta emergendo sempre più.
Gli strumenti attualmente disponibili infatti sono basati sull'uso della voce, e non consentono di ottenere una serie di ottimizzazioni che realizzerebbero un uso più efficiente delle risorse disponibili ed un approccio completamente elettronico ed automatizzato.
Tutto ciò diventa sempre più importante a causa della attuale tendenza all'incremento delle dimensioni delle aziende agricole, contesto nel quale sistemi avanzati di gestione dati e sincronizzazione assumono anche maggiore importanza.

6.2 Task 2 - Introduzione

Obiettivo di questo lavoro è stato progettare e realizzare almeno in versione preliminare un protocollo di comunicazione wireless dedicato al fleet management e alla sincronizzazioni di veicoli ed accessori nel campo delle macchine agricole, che è oggetto di proposta per un nuovo Standard Internazionale ISO.
In realtà durante lo svolgimento del lavoro si è reso necessario allargare la prospettiva notevolmente, alla ricerca delle prestazioni richieste, e di tale percorso nel presente documento è presente un resoconto abbastanza dettagliato. Infatti è stato necessario ragionare anche in termini di sistema di elaborazione e trasmissione, visto che il contributo alle prestazioni complessive dipende significativamente non solo dal protocollo, ma anche da tutti gli componenti del sistema di comunicazione.
Esistono vari vincoli da rispettare nel raggiungimento di un obiettivo quale quello di approvare un nuovo Standard Internazionale, alcuni di natura tecnica, ed altri appartenenti ad altri settori, ad esempio legislazioni differenti in vari paesi. Il presente documento concentrerà la sua attenzione sul versante tecnico, cercando però di non trascurare del tutto le altre questioni.
Allo scopo di incrementare la produzione realizzando sessioni di lavoro più efficaci in campo agricolo è utile riuscire a coordinare gruppi di macchine che lavorino cooperativamente per

ottenere lavorazioni parallele di colture e campi. Tutte le macchine sia di tipo eterogeneo che omogeneo all'interno del gruppo, sia che lavorino in serie che in parallelo, devono essere sincronizzate rispettando stringenti requisiti di sicurezza allo scopo di raggiungere gli obiettivi delle varie lavorazioni possibili.

Le specifiche per una sincronizzazione adeguata richiedono un protocollo di comunicazione wireless real-time che sia poco costoso da implementare, sicuro e con bassa latenza, allo scopo di assicurare un throughput minimo garantito e sufficiente a sincronizzare il lavoro e gli spostamenti delle macchine.

Di seguito saranno descritte nel dettaglio le caratteristiche del protocollo di trasmissione wireless quasi-isocrono basato sulla suite standard TCP/IP, ed in particolare su UDP, sviluppato utilizzando esclusivamente software ed hardware aperti, ed oggetto del Task 2 nel presente Dottorato. In particolare sono stati utilizzati nel corso del lavoro solo strumenti FLOSS per la parte software, ad esempio Linux come sistema operativo, e piattaforme aperte quali PC 'x86 compatibili' o embedded PC/104 per la parte hardware. Alcune piattaforme hardware proprietarie sono state utilizzate allo scopo di estendere il parco macchine testato. Il protocollo oltre che progettato ed implementato in una forma preliminare, è stato dunque testato su una casistica abbastanza ampia di sistemi, e in condizioni diverse grazie alla realizzazione di vari siti ed un laboratorio per i test. Per facilitare lo svolgimento dei test stessi è stata realizzata una procedura completamente automatizzata che consente di operare anche da remoto su un numero arbitrario di macchine. Il sistema di sviluppo è anch'esso distribuito e basato completamente su strumenti software FLOSS. Nel seguito le prestazioni rilevate saranno descritte da un punto di vista quantitativo e statistico. I test ottenuti suggeriscono che l'architettura proposta per il protocollo rappresenta una soluzione promettente per comunicazioni a bassa latenza per applicazioni mobili ed industriali, e, come già detto, sarà oggetto di proposta per un nuovo Standard Internazionale ISO.

6.3 Task 2 - Visione e futuro a lungo termine

Un breve cenno allo scenario in cui il lavoro oggetto della presente tesi è inserito, per meglio chiarire quale potrebbe essere la rilevanza di nuovi strumenti atti a favorire la cooperazione e la sinergia delle macchine da lavoro.

La disponibilità e l'utilizzo di sistemi di elaborazione dati sono divenuti pervasivi, facilitati dall'avanzamento della tecnologia e dall'abbassamento dei costi. Sistemi elettronici sono

ormai contenuti in quasi qualsiasi tipo di oggetto, anche dei più comuni, tanto che è impensabile immaginare un veicolo mobile senza elettronica a bordo.
Il passo successivo sarà quello di realizzare una 'collaborazione' e sincronizzazione di tutte le 'intelligenze sintetiche' a bordo dei veicoli in modo da consentire di assolvere a compiti altrimenti non realizzabili.
A tal scopo in tutto il mondo si stanno già aprendo scenari tecnologici che prevedono sempre di più veicoli e macchine autonome cooperative robotizzate (marine, terrestri, aeree), per svolgere compiti molto complessi, ad esempio in termini di difesa militare, di incremento di produttività delle attività agricole, di sicurezza ambientale e civile, di supporto a politiche ecologiche e controllo del territorio, di mobilità civile e trasporti commerciali.
E' molto interessante notare come ci sia di fatto una convergenza di molti e disparati settori verso scenari di questo tipo, che sicuramente rappresenterà un ulteriore fattore abilitante grazie allo scambio tecnologico.
Ciò ci spinge a pensare che con tutta probabilità simile sarà il futuro anche del settore agricolo, una visione in cui saranno presenti cluster di veicoli autonomi cooperativi ed efficienti, il cui programma di lavoro sarà aggiornato in tempo reale sulla base di informazioni provenienti da sistemi diversi di acquisizione delle informazioni ed anche direttamente da una analisi ambientale in tempo reale.

6.4 Task 2 - Il futuro a medio e breve termine

Concentrandoci ora sul settore agricolo possiamo più facilmente immaginare quale sarà il futuro a medio termine, ovvero una situazione simile a quella descritta in Figura 6.1:

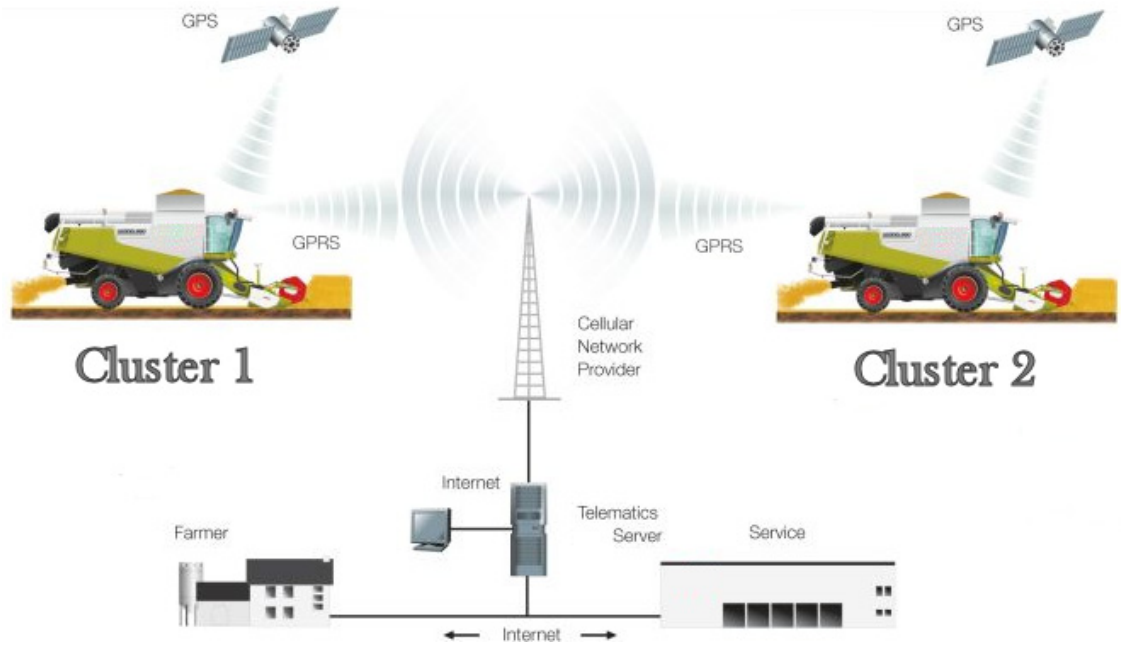


Figura 6.1: Lo scenario Agricolo a medio termine

Possiamo tranquillamente pensare che nel medio termine sarà completata la realizzazione di macchine agricole organizzate in cluster sincronizzati e cooperative. La guida non sarà completamente autonoma, ma con almeno un essere umano ai comandi supportato da sistemi di realtà aumentata (dei quali si vedono già degli esempi), che riescano ad aiutare gli operatori nelle sessioni di trattamento in base ad analisi su piante e terreno sia in ambiente aperto che in serre. Verranno inoltre sicuramente utilizzati dati provenienti da sistemi GPS per consentire l'individuazione ed il trattamento del terreno tenendo in conto delle sue caratteristiche puntuali, (di fatto ciò accade già) in modo da ottimizzare l'utilizzo di fertilizzanti ad esempio, oppure per modulare opportunamente l'intensità delle azioni meccaniche come l'aratura. Inoltre la generazione e comunicazione in tempo reale di dati di produzione e fleet management, trasmessi a sistemi di gestione tramite reti wireless e Internet, consentirà di ottimizzare varie fasi del processo produttivo, ed anche la logistica, in modo da incrementare la produttività dell'intera catena e minimizzare gli sprechi.

Volendo ragionare invece a più breve termine, e considerando uno scenario realizzabile nell'arco di pochi anni, arriviamo all'oggetto del lavoro svolto nel presente Dottorato, ovvero il controllo cluster di macchine.

I componenti necessari per realizzare tale obiettivo sono in fase avanzata di realizzazione, ad esempio esistono già sul mercato i primi sistemi di guida autonoma per macchine agricole, in fase di avanzata sperimentazione.

Per questo motivo sarebbe molto utile riuscire ad approvare in tempi brevi uno Standard Internazionale comprendente un protocollo real-time wireless short range che consenta di sincronizzare le macchine facenti parte del cluster, perchè a questo punto l'obiettivo sarebbe quasi raggiunto. Vediamo ora qualche esempio di cluster di veicoli collaborativi in ambito agricolo.

6.4.1 Cluster di veicoli collaborativi omogenei

In Figura 6.2 a titolo di esempio è mostrato un tipo lavorazione in cui è impiegato un cluster di veicoli collaborativi omogeneo:

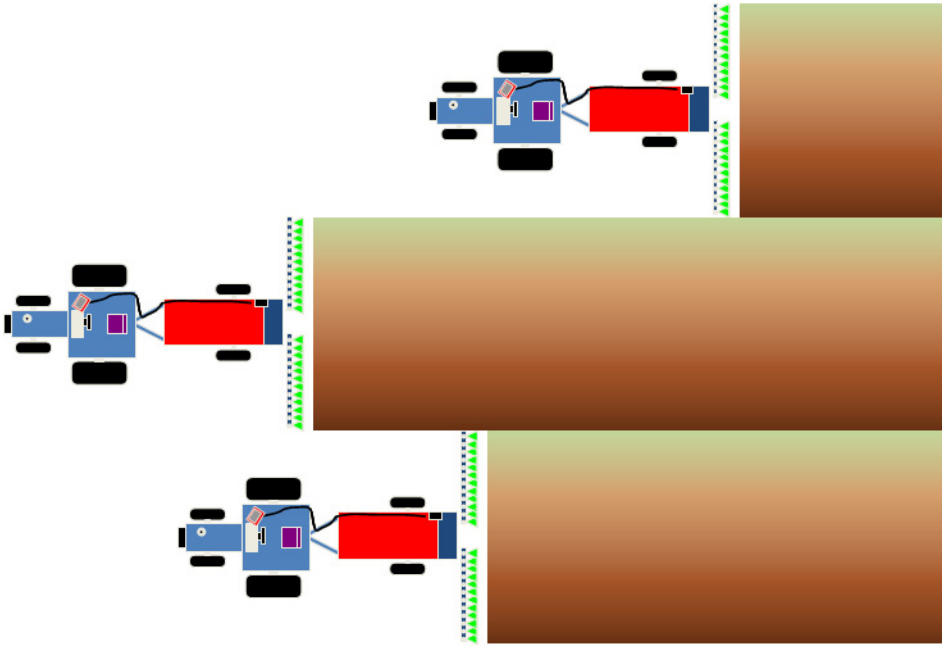


Figura 6.2: Lavorazione con gruppo di macchine omogenee

In figura sono mostrati un gruppo di trattori con relativi accessori irroratori per il trattamento parallelo di un campo che si suppone di notevole estensione. Avendo a disposizione la tecnologia necessaria si può immaginare che uno solo dei trattori sia guidato da un operatore,

e gli altri collaborino alla lavorazione grazie ad una sincronizzazione real-time. Tale scenario potrebbe sembrare avveniristico, ma in realtà è dietro l'angolo. Ad esempio, come già accennato, alcuni sistemi di guida assistita da remoto iniziano ad affacciarsi sul mercato.

Per meglio comprendere quali sono le necessità di comunicazione e sincronizzazione richieste nel caso di una lavorazione del tipo di quella citata analizziamo ora quali potrebbero essere alcune delle fasi richieste al completamento della attività:

- Per prima cosa il gruppo di macchine omogenee avviamo una sessione di comunicazione sfruttando una rete Wi-Fi ad esempio. Dal punto di vista del protocollo di comunicazione il ruolo di Master nella sessione di lavoro viene attribuito alla macchina dotata di operatore.
- La macchina Master dunque avvia una sessione di comunicazione durante la quale lo stato della macchina stessa ed altri dati riguardanti il compito che deve essere eseguito vengono distribuiti alle altre macchine collaboranti.
- Durante la lavorazione vera e propria dati riguardanti le posizioni relative, velocità, accelerazioni dovranno essere scambiati per regolare opportunamente le traiettorie delle macchine pilotate, e per mantenere tutto il sistema di lavorazione in condizioni di sicurezza.
- A fine lavorazione similmente saranno trasmesse tutte le informazioni necessarie perché il cluster di veicoli torni ordinatamente in condizione di riposo.

I dati da trasmettere sono ad esempio posizione GPS della macchina Master e informazioni riguardanti le caratteristiche dal campo da trattare. Altri dati potrebbero riguardare la configurazione da assumere per consentire agli altri veicoli di regolare la propria posizione durante la lavorazione per evitare di sovrapporre le zone trattate, oppure al contrario di lasciarne alcune senza trattamento. Uno dei dati sarebbe sicuramente legato alla dimensione degli accessori stessi collegati ai trattori, che sono indispensabili a definire quale debba essere la posizione relativa dei trattori nel campo.

Oltre a ciò la dinamica della trasmissione dati deve tener conto in tempo reale delle dinamiche 'fisiche' delle macchine, ovvero cambio di direzione e velocità, o ancora più precisamente anche il cambio di direzione degli irroratori, che potrebbe dover essere considerato anche con una granularità di poche centinaia di millisecondi.

A causa di tali specifiche una quantità garantita di informazioni deve essere trasferita dalla macchina Master a quelle circostanti.

In altre parole una volta che la sessione è stata avviata deve essere garantito un flusso continuo di informazioni in grado di creare una lavorazione cooperativa a controllo distribuito, facendo

in modo che le macchine che vi partecipano rimangano collegate durante tutta la durata delle operazioni programmate.
Le dinamiche coinvolte in un sistema di questo tipo sono un dato noto, e richiedono un throughput dell'ordine di grandezza dei 500 byte ripetuto a passi di 50 o 100 millisecondi per ogni macchina, in modo da poter trasmettere stato e comandi operativi, come specifica minima che consenta la sincronizzazione delle macchine.
In realtà in una lavorazione complessa possono essere definiti vari compiti che abbiano simili requisiti in termini di real-time. Di più, a causa delle svariate funzionalità implementate in una tipica unità di controllo installata su macchine agricole, possono essere definiti vari tipi di traffico dati, ognuno con associati differenti livelli di prestazioni real-time e sicurezza richiesti.
Una macchina di alto livello può già ora essere equipaggiata con un sistema di fleet management che consente un collegamento continuo con l'azienda agricola per controllo delle sessioni di lavoro in corso, riprogrammazione dinamica delle stesse, e conservazione dei dati di produzione e lavorazione; tipicamente vengono utilizzate allo scopo connessioni GPRS/ UMTS e poi Internet come tramite.
Il programma di lavorazione, i dati di fleet management e così via possono però essere considerati dati a priorità inferiore e non real-time, anche se il traffico ad essi associati non può essere trascurato, perché verrà gestito tramite le stesse risorse nel sistema, magari anche utilizzando protocolli già esistenti come TCP. Si può comunque immaginare che un nuovo protocollo preveda anche differenti livelli di priorità associati a diversi tipi di traffico, ed anche differenti strutture dei pacchetti che vengano incontro alle diverse necessità di comunicazione. Tutto ciò sarà approfondito nei paragrafi successivi.

6.4.2 Cluster di veicoli collaborativi eterogenei

Un esempio di lavorazione effettuata con macchine eterogenee è presente in Figura 6.3. In alto a sinistra in figura è presente uno schema che evidenzia meglio il tipo di lavorazione nella foto. In questo caso consideriamo la lavorazione combinata di una mietitrebbia che raccoglie dal campo un prodotto agricolo e lo trasferisce automaticamente nel rimorchio trascinato da un trattore. La mietitrebbia deve rimanere in contatto con il rimorchio ottenendo la misura del livello del prodotto nel rimorchio ed anche la posizione del braccio trasportatore in relazione alla lunghezza del rimorchio stesso. La mietitrebbia dovrebbe comandare il movimento del

trattore che trascina il rimorchio, in modo che rimanga sincronizzato, ad esempio accelerando o rallentando opportunamente.

Figura 6.3: Lavorazione con gruppo di macchine eterogenee

A rimorchio pieno potrebbe notificarlo al sistema di fleet management e al trattore e smettere di scaricare nel rimorchio il prodotto raccolto. Per fare ciò si può misurare la distanza tra un sensore sul braccio ed il livello del prodotto nel rimorchio, ad esempio. Conoscendo i livelli minimo e massimo del prodotto nel rimorchio, l'altezza del rimorchio, ricevuta dal trattore, e l'altezza del braccio acquisita direttamente tramite sensori, il sistema potrebbe determinare in tempo reale il livello di riempimento ad una certa posizione all'interno del rimorchio.

Pertanto la tipica lavorazione sul campo richiede un Master che comunica parametri e comandi ad un o più Slave. Il compito degli Slave è quello di seguire il Master con direzione e velocità definite in real-time. Perciò tutte le macchine Slave devono essere provviste di un sistema di guida parzialmente (in futuro completamente) autonomo, o al minimo, con una interfaccia che consenta la guida assistita. Il sistema di comunicazione tra le macchine deve fornire una trasmissione in real-time, in modo da riuscire ad attivare una lavorazione a controllo distribuito, ed inoltre assicurarsi che le macchine rimangano collegate e sincronizzate nel corso delle operazioni programmate.

Come si può vedere in foto attualmente la lavorazione richiede due operatori, se le macchine fossero sincronizzate in cluster un solo operatore sarebbe necessario.

In entrambi i casi esaminati di collaborazione tra macchine agricole le informazioni necessarie alle lavorazioni sono simili, ed in realtà con un opportuno protocollo sarebbe possibile scambiare tutte le informazioni necessarie per vari tipi di compiti, purché vengano rispettate alcune specifiche in termini di latenza, throughput e sicurezza. L'approfondimento di tali questioni sarà oggetto dei prossimi paragrafi.

6.5 Task 2 - Premesse sul contesto

Il nuovo protocollo ideato è stato concepito per funzionare su macchine agricole, dove esistono già altri protocolli e tecnologie ampiamente utilizzate. Ad esempio reti CAN - ISO 11898 come strato di comunicazione a basso livello nello stack ISO/OSI, e a più alto livello SAE J1939 e ISO 11783 sono le piattaforme standard per comunicazione intraveicolare, già utilizzate per controllo macchina e relativi accessori agricoli.
In considerazione delle funzionalità che dovrebbero essere fornite dal nuovo protocollo proposto è fortemente consigliabile, se non addirittura richiesta, una compatibilità con la struttura di comunicazione adottata da CAN, in modo da facilitare ad esempio le attività di un router o gateway o altro genere di controllo elettronico similare responsabile del trasferimento e dell'instradamento di informazioni da e per una rete cablata in standard CAN.
In conseguenza di questo scenario sono quindi nel seguito state scelte alcune definizioni, ed adottate alcune scelte, in modo da mantenere una struttura della comunicazione e della rete quanto più possibile vicina e quindi compatibile con i protocolli di comunicazione ISOBUS e SAE J1939.
D'altro canto, al fine di realizzare un protocollo di comunicazione universalmente disponibile e accettato, una soluzione efficace è l'utilizzo della suite di protocolli TCP/IP. Tale suite è estremamente diffusa, a partire dal mondo dei personal computer, per continuare in quello dei sistemi embedded, ed è utilizzato in moltissimi ambiti tra cui quello industriale, automotive ed anche agricolo per comunicazioni long range. Sarebbe pertanto una scelta naturale e porterebbe come conseguenza una immediata compatibilità con lo standard per comunicazione di rete digitale più diffuso e larghissimamente utilizzato. Ciò porta anche a poter scegliere hardware praticamente senza ulteriori limiti tra quanto già disponibile sul

mercato, in base semplicemente al campo di applicazione. Il TCP/IP infatti è innato su qualsiasi dispositivo, a partire dai microcontrollori in uso embedded fino ai sistemi più potenti. Questa disponibilità così ingente e diffusa facilita la creazione e l'adozione di un nuovo standard. Come vedremo nei prossimi paragrafi però la suite TCP/IP standard senza modifiche non soddisfa tutti i requisisti richiesti.

6.6 Task 2 - Una scelta 'Open'

In questo paragrafo vengono richiamati alcuni concetti che permeano tutto il lavoro svolto nel presente Dottorato. Le nozioni di base sul software libero sono già state introdotte ad esempio nel quarto capitolo al paragrafo 4.7.1 e seguenti, e non vengono qui ripetute. Similmente è stato già descritto in precedenza uno standard hardware aperto, ovvero PC/104 e derivati ancora nel quarto capitolo paragrafo 4.11.2, per cui si introdurranno solo gli elementi di diversità dei nuovi hardware utilizzati rispetto ai precedenti, senza tornare sulla descrizione dello standard. In questa parte però sono richiamati alcuni effetti e vantaggi derivanti dall'utilizzo di software FLOSS ed hardware aperto nella 'costruzione' di un nuovo Standard Internazionale, che si è ritenuto utile approfondire e riassumere.
Per quanto riguarda la creazione di un nuovo Standard ISO infatti una soluzione possibile è quella di utilizzare esclusivamente software Open-Source e hardware standard già prodotto, quindi non dedicato o progettato appositamente per l'applicazione.
La scelta del software utilizzabile è ricaduta immediatamente nell'ambito dell'Open-Source per molteplici motivi:

- Il software Open-Source è utilizzabile da chiunque e il suo uso richiede costi sensibilmente inferiori rispetto a soluzioni proprietarie
- La proprietà intellettuale non è un problema, uno standard 'costruito' tramite software Open-Source è, per sua natura, 'aperto'
- La disponibilità di codice già sviluppato, ampiamente documentato e liberamente modificabile semplifica notevolmente il lavoro
- Il software Open-Source è alla base di reti come Internet e quindi vi sono molti strumenti disponibili che sono abbondantemente testati e quindi affidabili

- In linea generale il mondo Open-Source è vicino agli Standard, e tipicamente si fa riferimento a questi ogniqualvolta è possibile, evitando soluzioni chiuse e proprietarie. Ciò favorisce l'armonizzazione con Standard preesistenti.

Per quanto riguarda invece l'hardware si utilizzano componenti standard e già in commercio invece di progettarli appositamente, per le seguenti ragioni:

- Ampia disponibilità di componenti, facilmente reperibili e certificati, ed indipendenza da un singolo produttore garantita
- Tecnologie comuni largamente usate, non solo in campo agricolo, e quindi con maggiore disponibilità e con bassi costi
- Uso di una classe di tecnologie in cui ci sono ingenti, continui e variegati investimenti in ricerca e sviluppo. Ciò le rende realmente competitive in termini di costi riferiti alle prestazioni ottenibili
- La capillare e massiva adozione significa maggiore affidabilità
- La grande diffusione implica spesso la armonizzazione con le norme internazionali. Un esempio è lo standard 802.11 che, nelle sue diverse declinazioni, è utilizzato praticamente ovunque, anche se a livello locale si possono trovare piccole differenze.

L'adozione di strumenti e prodotti a basso costo, di ottima qualità e grande diffusione anche se non immediatamente legata allo Standard, ne semplifica però grandemente l'adozione. Viene a formarsi infatti un humus fertile ed a bassa difficoltà di penetrazione che consente un facile sviluppo ed una rapida adozione. Al contrario un ambiente proprietario, con strumenti e prodotti ad alto costo e bassa diffusione sono spesso stati causa del fallimento di nuovi standard nel fondamentale momento della loro adozione.

6.7 Task 2 - Protocolli di riferimento

Iniziamo ora a ragionare sugli strumenti utili alla costruzione del nuovo protocollo già accennati nei paragrafi precedenti, allo scopo di evidenziarne gli aspetti utili e le criticità.
Il protocollo richiesto da una comunicazione a corto raggio dedicata alla sincronizzazione e collaborazione di macchine è sicuramente caratterizzato da requisiti in termini di real time e

sicurezza delle comunicazioni, a causa della necessità di informazioni affidabili e di un massimo ritardo di trasmissione minimizzato e controllato.

Problemi simili sono stati analizzati ed in alcuni casi risolti in altri campi di applicazione, ad esempio per la trasmissione di segnali audio e video, oppure nel settore automotive, ma tra le soluzioni esistenti è molto difficile trovarne una che soddisfi contemporaneamente tutti i requisiti del caso oggetto di studio. Ad esempio messaggi di altro genere non devono disturbare la comunicazione real-time tra i partecipanti di una sessione di lavoro. Per questo motivo una gestione delle priorità nei flussi di dati complessivi in ingresso ed uscita deve essere progettata, in modo da definire ad esempio delle classi di traffico real-time definite in base al massimo ritardo ammissibile per i dati da trasferire appartenenti ad una specifica classe. I protocolli analizzati in letteratura sono spesso basati sulle caratteristiche della suite TCP/IP, molti funzionano su piattaforme hardware Intel based tipicamente derivate dai PC, e che supportano sistemi operativi completi e personalizzabili come Linux, ma che sono privi anche a livello di sistema delle caratteristiche di real-time che sono invece tipiche dei sistemi embedded. Anche se altri protocolli come RTP **[9]** sono stati progettati dopo alcuni anni rispetto alla suite TCP/IP per trasferimento di dati real-time, nessuno prevede le richieste funzionalità di sicurezza. Infatti TCP/IP anche se dotato di funzionalità avanzate quali ad esempio affidabilità, controllo di flusso, concetto di connessione , è stato originariamente progettato per applicazioni senza reali vincoli di real-time; per questo motivo presenta alcuni aspetti critici nella gestione del traffico riguardo alla latenza dei pacchetti.

L'idea di utilizzare come base TCP/IP, per sfruttare le sua alta diffusione e compatibilità con tecnologie e protocolli di trasferimento dati richiede dunque ulteriori analisi, a livello degli strati più bassi della pila ISO/OSI, per assicurarsi del corretto comportamento del sistema che si vuole implementare. Di converso il protocollo UDP presenta alcune caratteristiche interessanti, ma la sua impostazione 'best effort' non garantisce la richiesta affidabilità e il livello di sicurezza richiesto da applicazioni tipiche del mondo agricolo, facendo dunque emergere la necessità di una strategia di controllo di livello superiore che sarà nel seguito descritta.

I protocolli di trasporto sinora trattati risolvono il problema della trasmissione dei dati, ma si potrebbe porre anche la questione di una rete la cui topologia non è statica, ma varia durante una sessione. La gestione di problematiche di questo tipo è affidata a protocolli che gestiscono le cosiddette reti wireless adhoc (mesh networks). In letteratura sono presenti molti tipi di protocolli, ed è un campo di ricerca molto attivo. Nel corso del presente Dottorato la questione

è stata affrontata brevemente a scopo di documentazione su problemi che potrebbero essere contingenti. Visto però che molti altri gruppi di ricerca sono impegnati su questo fronte, e che è un ramo differente rispetto al corrente lavoro, si è fatta la scelta di citare brevemente la questione e inserire solo alcuni cenni alle tipologie di algoritmi per consentire un inquadramento del problema. In bibliografia sono presenti vari riferimenti che consentono di approfondire la questione, si veda ad esempio **[10]**, **[11]**, **[12]**, **[13]**.

Riassumendo molto la questione comunque il problema principale dal punto di vista del protocollo oggetto del presente documento è la latenza introdotta, ed anche il fatto che deve essere trasmesso del traffico ulteriore per la gestione dinamica della topologia della rete. Tale traffico però può essere classificato come a più bassa priorità e non dovrebbe causare rilevanti problemi al nuovo protocollo, come si potrà comprendere osservando i test effettuati.

Vediamo un breve elenco delle tipologie di protocolli per reti adhoc esistenti:

- Pro-active (table-driven) routing: questo tipo di protocolli mantiene liste aggiornate di destinazioni e le route associate distribuendo periodicamente tabelle di routing ai vari nodi delle rete. I maggiori inconveniente di algoritmi di questo tipo sono la notevole mole di dati necessari per l'aggiornamento e la lenta reazione a modifiche della topologia o perdita di contatto radio.
- Reactive (on-demand) routing: questo tipo di protocolli scopre una route inviando continuamente in rete pacchetti di 'Richiesta Route'. Gli inconvenienti principali sono l'alta latenza introdotta nel scoprire nuove route e il tasso di invio eccessivo di pacchetti che può sprecare inutilmente risorse ed addirittura portare la rete a saturazione.
- Flow-oriented routing: questo tipi di protocolli cerca una nuova route solo su richiesta, partendo solo dalle tabelle di routing disponibili al momento. Lo svantaggio maggiore consiste nel fatto che può servire molto tempo nella ricerca di route sconosciute rispetto ai tempi tipicamente richiesti per la trasmissione.
- Hybrid (both pro-active and reactive) routing: questo tipo di algoritmi cerca di combinare i vantaggi del routing proactive e reactive. Il routing viene inizialmente avviato con route cercate automaticamente all'atto della inizializzazione, ed in seguito le richiesta di raggiungere nodi ulteriori attivati in seguito viene eseguita solo su richiesta. Anche questi algoritmi però hanno svantaggi, rimanendo comunque rigidi.
- Adaptive (situation-aware) routing: è simile al caso precedente, ma la scelta del tipo di azione è legata all'uso di specifiche metriche.

Dopo questa rapida carrellata partiamo ora da uno dei componenti fondamentali, che è la suite TCP/IP, o più precisamente i protocolli TCP, UDP e IP. Vedremo che a patto di alcune modifiche e aggiunte con tali componenti sarà possibile costruire una proposta che soddisfi i requisiti richiesti. In questa parte ci concentreremo su uno degli aspetti fondamentali delle prestazioni richieste al nuovo protocollo, ovvero una latenza di trasmissione più bassa possibile. Dapprima saranno analizzati i protocolli di comunicazione disponibili, facenti parte della suite TCP/IP ed in seguito un'altra delle fonti principali di latenza in sistemi di calcolo complessi, ovvero il sistema operativo ed il software applicativo. Tutti questi elementi infatti concorrono a determinare le prestazioni complessive del sistema. Nel seguito saranno poi presentate tutte le soluzioni messe in campo per mitigare od eliminare i problemi considerati.

6.7.1 TCP, UDP e IP

Avendo come riferimento le caratteristiche e le prestazioni elencate nei paragrafi precedenti, al fine di realizzare un protocollo di comunicazione universalmente disponibile e accettato, la soluzione più efficace è sicuramente quella di mantenere quanto più possibile la compatibilità con la suite di protocolli TCP/IP, come già anticipato nel paragrafo introduttivo. È necessario però sottolineare che il protocollo TCP/IP senza nessuna modifica non soddisfa le specifiche performance richieste dallo Standard per il mondo agricolo che si vorrebbe creare, semplicemente perché il TCP è stato creato per scopi differenti. Per questa ragione alcuni aspetti del TCP verranno brevemente discussi in seguito e saranno proposte alcune modifiche che potrebbero portare comunque all'uso di un protocollo modificato ma perfettamente compatibile. In realtà in una fase preliminare era stata anche valutata la possibilità di usare il TCP/IP 'standard' per realizzare lo Standard ISO oggetto del presente Task 2.
Per chiarificare alcuni aspetti è di utilità la seguente tabella, che esplica la relazione tra lo stack TCP/IP e lo standard Open System Interconnection (OSI).

<table>
<tr><th>ISO/OSI</th><th>TCP/IP</th></tr>
<tr><td>7. Application</td><td rowspan="3">Application
(telnet, FTP, SMTP, DNS, HTTP, etc.)</td></tr>
<tr><td>6. Presentation</td></tr>
<tr><td>5. Session</td></tr>
<tr><td>4. Transport</td><td>Transport (TCP and UDP)</td></tr>
<tr><td>3. Network</td><td>Network (IP, ARP, etc.)</td></tr>
<tr><td>2. Data link</td><td rowspan="2">Host - Network</td></tr>
<tr><td>1. Physical</td></tr>
</table>

Tabella 6.4: Lo stack ISO/OSI

Il protocollo TCP/IP può essere interpretato come un insieme di livelli che, cooperando tra loro, realizzano e mantengono la comunicazione. È buona norma che i livelli più alti contengano implicitamente i più bassi, nascondendo così la complessità generale. Questo approccio, se da un lato è più chiaro, dall'altro introduce alcune limitazioni, in quanto la separazione va a ledere un sistema che linearmente sarebbe più logico.
I primi due livelli, Data-link-layer and Physical, che sono legati all'hardware, verranno trattati successivamente, ora sono presi in analisi i livelli più alti.
Nel TCP/IP i tre livelli più alti (Application, Presentation and Session) della pila ISO/OSI sono tipicamente unificati e fusi in un unico livello, l'Application Layer. Questo livello è quindi da considerarsi al di fuori della discussione, nel senso che i livelli che devono essere definiti dalla norma ISO sono i primi quattro livelli della pila ISO/OSI, essendo i tre strati di livello superiore in genere 'dipendenti dall'applicazione'.
Il quarto livello, Transport, è l'interfaccia naturale della maggior parte delle applicazioni. I protocolli direttamente visibili a questo livello sono i protocolli TCP e UDP. In realtà, nonostante le necessità di molte applicazioni sono soddisfatte dal TCP o UDP, è possibile un approccio diretto al livello IP, ovviamente rinunciando a tutti i servizi forniti dagli strati di livello superiore, e rinunciando alla regola di nascondere gli strati di basso livello da parte dei livelli superiori. L'accesso dell'applicazione alla risorsa di comunicazione viene solitamente realizzato dalle librerie 'Network Sockets Application Programming Interface' o altrimenti dette librerie 'Network Sockets API'. Va notato che l'implementazione delle Network Socket API è disponibile per molti linguaggi di programmazione e per molti ambienti di sviluppo, mentre l'accesso al livello tre (Network Layer) non è altrettanto supportato. Per la realizzazione dello Standard ISO oggetto della presente trattazione, che vorrebbe integrarsi con il TCP/IP si deve scegliere se utilizzare come protocollo di trasporto il TCP o l'UDP o anche l'accesso diretto all'interfaccia IP, se le due precedenti soluzioni si rivelassero non sufficientemente performanti. Iniziamo ad analizzare il protocollo TCP.

6.7.2 Transmission Control Protocol (TCP)

Il Transmission Control Protocol (TCP) è un protocollo di trasporto che ha però a disposizione maggiori funzionalità rispetto ad altri protocolli simili ed anche della stessa 'famiglia' (affidabilità, controllo del flusso, riordinamento pacchetti, creazione di sessioni) ma il suo

ambiente d'elezione è in reti dove stringenti restrizioni di tempo non esistono ed una comunicazione con un real-time ad alte prestazioni non è di fondamentale importanza.
Il TCP può essere in teoria usato in applicazioni sensibili alla latenza, ma andrebbero considerati diversi fattori. Ad esempio TCP rimane a bassa latenza solo se i ricevitori del traffico sono sempre 'attivi', e se il canale non è congestionato, in pratica la latenza è bassa finché 'va tutto bene' e non si riscontrano significativi ostacoli.
È facile pensare al TCP come ad un protocollo di trasporto che mantiene l'affidabilità della comunicazione, ma l'architettura deve garantire l'accesso equo a tutti gli utenti. Se fosse possibile a tutti gli utenti accedere a piacimento, si potrebbero creare situazioni di congestione, dove la rete non riesce più a gestire adeguatamente il carico di dati.
Quindi il TCP applica una sorta di arbitraggio sulla trasmissione dei pacchetti, registrando quelli in attesa di essere spediti in un buffer. A ciò è dovuto il nome Transmission Control Protocol, in quanto di fatto viene attuato un controllo sulla trasmissione.
Quindi il TCP introduce latenza per mantenere stabile la rete. Tale comportamento può essere adeguato o addirittura preferibile in alcuni contesti e applicazioni, ma le strategie adottate dovrebbero essere opposte per applicazioni sensibili alle latenze.
Sarebbe comunque possibile migliorare il comportamento del protocollo TCP, mediante alcune modifiche e/o configurazione di parametri dello stesso. Ad esempio:

1. Usare una bufferizzazione intelligente o socket non bloccanti per poter 'droppare' alcuni pacchetti a bassa priorità in situazioni di congestione pur di mantenere una bassa latenza
2. Disabilitare l'algoritmo di Nagle attivando l'opzione TCP_NODELAY in setsockopt (anche se questa modifica realizza un miglioramento per quanto concerne la latenza ma a scapito dell'efficienza)
3. Impostare un'alta priorità al traffico che deve avere bassa latenza (QoS)
4. Mantenere il buffer delle socket ad una grandezza minima cosicché non vi sia introduzione di latenza a causa della bufferizzazione ma piuttosto una perdita di dati.

6.7.3 User Datagram Protocol (UDP)

UDP e' un protocollo di trasporto che garantisce solo alcune funzionalità di base (senza connessione, con un CRC più leggero). Il suo utilizzo però richiede una migliore caratterizzazione dei livelli alti per garantire alcune funzionalità tipiche del TCP come ad esempio l'utilizzo di messaggi di ACK per trasmissioni corrette (non previste nell'UDP). Il

protocollo UDP sembra una buona scelta data l'ampia disponibilità e una struttura piuttosto semplice ed inoltre facilmente modificabile per adeguarsi ad un nuovo Standard.

6.7.4 Comparazione tra UDP e TCP

Sono qui riportate alcune valutazioni sintetiche comparative tra i due protocolli di trasporto, riferite alle principali caratteristiche di UDP, al fine di evidenziarne le differenze.

1. Nessuna connessione. TCP usa un metodo di inizializzazione della connessione prima di cominciare a trasferire dati mentre UDP si limita a trasmetterli, senza nessuna informazione preliminare. Quindi UDP non introduce ritardi all'inizio della connessione. HTTP ad esempio usa TCP per stabilire le connessioni, dato che l'affidabilità è importante per le pagine web testuali. Ciò però si traduce in una esperienza comune, ovvero nel fatto che a causa del ritardo introdotto dal TCP, la visualizzazione delle pagine possa essere ritardata in funzione del livello di congestione della rete.
2. Nessuna informazione sullo stato della connessione. Il protocollo TCP mantiene l'informazione sullo stato della propria connessione. Vi sono quindi buffer di ricezione, invio di parametri per il controllo della congestione, pacchetti numerati in modo sequenziale ed ack. Inoltre i buffer non sono svuotabili a piacere dall'applicazione. Dato che tutte le caratteristiche introdotte da TCP non sono fondamentali per uno Standard del tipo oggetto della presente trattazione, questo caso la scelta tende verso UDP, protocollo che non tiene traccia a priori di nessuna di queste informazioni. Se fosse necessario sarebbe possibile aggiungere la gestione delle informazioni richieste.
3. Peso dell'Header. Il protocollo TCP aggiunge al pacchetto un header di 20 Byte, mentre il protocollo UDP soltanto di 8 Byte.
4. Unregulated send rate. Il controllo di congestione del TCP regola automaticamente la frequenza di invio dei pacchetti, per evitare congestione della rete. Questa regolazione può avere un forte impatto sulle prestazioni di un sistema real-time che a fronte di una perdita di pacchetti informativi richiede il mantenimento certo di una frequenza di invio degli stessi. Parallelamente bisogna considerare che la frequenza alla quale UDP invia i pacchetti è dipendete solo dalla velocità con la quale vengono generati dall'applicazione. Inoltre bisogna tenere presente che non tutti i pacchetti inviati dalla sorgente arriveranno a destinazione a causa o di buffer overflow o di problemi legati alla trasmissione wireless.

Bisognerà quindi implementare un sistema di recupero dei pacchetti persi, probabilmente anche diversi in funzione del tipo di applicazione richiesta, ma diversi da quello implementato in TCP dato che lo scopo del nuovo protocollo è differente da quello del TCP.

6.7.5 Accesso diretto al protocollo IP

Il protocollo UDP aggiunge meno funzionalità ed un CRC più leggero, nel campo agricolo però è necessario utilizzare un CRC più pesante. Per questo si potrebbe considerare un approccio direttamente al livello IP. Altro vantaggio dell'accesso diretto invece dell'incapsulamento in UDP potrebbe essere quello di utilizzare un protocollo completamente nuovo, e riconoscibile banalmente dai dispositivi di rete, al fine di implementare più facilmente alcune caratteristiche. Ad esempio priorità dei pacchetti.
Usando il protocollo UDP di converso sarebbero direttamente utilizzabili svariate funzionalità sviluppate e pronte ed un interfacciamento standard con il livello di applicazione, oltre probabilmente ad una migliore compatibilità con la suite di riferimento e la base hardware già installata. Un altro lato da considerare è la maggior sicurezza che l'utilizzo di UDP sia supportato da un maggior numero di librerie software, mentre il supporto diretto allo stack IP può non essere garantito, ed anzi spesso non è presente.
Per questo sarebbe preferibile usare il protocollo UDP piuttosto che l'accesso nativo che verrebbe considerato solo se UDP non consentisse implementare tutte le specifiche per qualche limitazione. In seguito ad una fase di test ormai articolata si può dire comunque che sinora problemi che impediscano l'utilizzo di UDP non sono emersi, anzi sinora le prestazioni mostrare lasciano ben sperare sul suo utilizzo per la definizione del nuovo Standard.

6.8 Protocolli scelti

Considerando i ragionamenti e le analisi effettuate nei paragrafi precedenti, cercare di controllare la latenza usando il protocollo TCP sembra essere un'impresa difficile. Tale protocollo infatti implementa una serie di strategie atte a rendere molto robusta la connessione a discapito della latenza.
Una buona base di partenza può invece essere il protocollo UDP opportunamente configurato ed eventualmente modificato in modo da minimizzare la latenza introdotta. Varie azioni possono essere intraprese, ad esempio si può ridurre la dimensione dei buffer utilizzati per la trasmissione, verificare i ritardi introdotti nell'invio di pacchetti dal sistema operativo,

generare e trasmettere pacchetti di dimensioni tali da non dover essere frammentati, o ancora meglio abbastanza piccoli da rientrare sicuramente nei buffer previsti da tutta la catena di trasmissione, ancora minimizzare il numero di hop.
Oltre a ciò nei paragrafi successivi si considererà la questione della latenza da un punto di vista più ampio, legato ad una logica di sistema, che nella pratica è il contesto nel quale viene determinata la latenza rilevante ai fini pratici. Vedremo come agire per minimizzare anche le altre cause di latenza, che seppur non direttamente legate al protocollo possono comunque esserlo indirettamente, e comunque concorrono al risultato complessivo.
Vedremo come strumenti per migliorare le prestazioni potrebbero essere ad esempio l'aumento della priorità del processo responsabile della comunicazione al massimo possibile o l'utilizzo di un sistema operativo real-time dove ogni messaggio viene gestito dalla CPU in tempi estremamente ridotti.
In ogni caso per verificare se il protocollo UDP è davvero un buon candidato come base per elaborarne uno nuovo e più sofisticato, si è proceduto nel corso del lavoro con una articolata serie di test, realizzati con una strumentazione appositamente allestita e utilizzando una serie di software sviluppati ad-hoc, che saranno presentati e discussi nel seguito.

6.9 Latenza: incidenza del Sistema e del Software

Il sistema operativo installato su una macchina contribuisce alla latenza totale del sistema di comunicazione, stessa cosa accade anche per il software installato su un sistema embedded (firmware). Ovviamente anche il software applicativo può incidere sulle prestazioni totali, in realtà di ciò si tiene normalmente conto nel progetto e realizzazione di applicazioni dedicate al mondo agricolo ed alla controllistica in generale. Infatti questa classe di applicazioni richiede una progettazione specifica che non tenga conto solo delle funzionalità richieste, ma anche e attentamente dei tempi di esecuzione.
Va fatto notare che il software Open-Source essendo 'libero' in ogni sua parte è maggiormente modificabile ed adattabile alle condizioni di lavoro richieste. Ad esempio con Linux c'è la possibilità di configurare vari parametri dello scheduler del sistema operativo, o addirittura di riscriverlo.
Per quanto riguarda la scelta del sistema operativo potremmo decidere di utilizzarne uno real-time oppure di normale utilizzo, ovvero mainstream. Quest'ultimo ha sicuramente vantaggi in termini di diffusione e supporto hardware ad esempio, ma anche per l'esistenza di un

vastissimo parco di applicazioni già pronte e di immediato utilizzo a costo zero, ed anche integrabili in nuove applicazioni essendo anch'esse a sorgente aperto. Un sistema operativo real-time, nel nostro caso, parrebbe invece più indicato per raggiungere le performance richieste. Tipicamente però la scelta di sistemi operativi stessi e applicazioni specificamente concepite per il real-time si riduce notevolmente, e molte soluzioni tendono ad essere proprietarie. Nel seguito sarà dato ampio risalto alla questione e alle scelte effettuate.

6.9.1 Sorgenti di latenza

Come già accennato diversi fattori contribuiscono alla latenza totale della trasmissione: il protocollo di trasmissione, il sistema operativo e più in generale tutta la latenza legata al software del sistema di comunicazione; anche l'hardware stesso può generare ritardi nella esecuzione di alcuni compiti, od in certi momenti.
La latenza totale del sistema è data da diversi contributi e quindi si potranno avere diverse latenze. Ad esempio generate da differenti lunghezze nei messaggi da trasmettere o da improvvisi impegni della cpu. L'obiettivo principale è cercare di mantenere sempre bassa la latenza, anche in presenza di grossi carichi di lavoro e un modo è quello di analizzare tutte le fonti di latenza del sistema e cercare di ridurle singolarmente e complessivamente. Alcune fonti di latenza impattano su ogni messaggio, altre solo su alcuni. Alcune fonti sono variabili, altre sempre fisse. Le componenti fisse formano un limite sotto al quale non si può abbassare la latenza. Nel seguito allo scopo di meglio inquadrare la questione è presente una lista di diverse fonti di latenza tipiche dei sistemi di trasmissione ed elaborazione dati, da quella più importante a quella il cui contributo alla latenza è minore, tenendo sempre presente però che in ogni sistema i singoli contributi alla latenza complessiva possono variare.

- Intermediatori. Una fonte di latenza può essere la presenza di 'intermediatori', componenti responsabili della comunicazione tra due entità che ovviamente aggiungono latenza. Essi possono essere ad esempio una connessione USB su una scheda wireless, o l'esecuzione di programmi che passano da User Mode a Kernel Mode e viceversa, per cui viene introdotta latenza per il context switching.
- Garbage Collector. Una delle comodità degli odierni linguaggi di programmazione come Java o C# risiede nel fatto di non doversi preoccupare di tenere traccia e di liberare memoria occupata da parti di codice inutilizzato. Ciò accade grazie alla presenza di quello che viene

chiamato Garbage Collector, ovvero un processo che si occupa di liberare la memoria automaticamente ad intervalli di tempo intermittenti. Ma questa esecuzione casuale del Garbage Collector introduce una latenza variabile, dato che nel momento di esecuzione viene fermata l'esecuzione di altri processi in corso. Alcuni linguaggi permettono che l'esecuzione degli altri processi continui anche in presenza del Garbage Collector. Generalmente il Garbaging avviene infrequentemente ma quando si verifica può generare una latenza sensibile. Dato che tutti i dati in arrivo durante il Garbaging vengono messi in attesa in una ideale fila, meglio usare un linguaggio di programmazione come C, che non presenta questo problema, anche se costringe a gestire manualmente la memoria.

- Ritrasmissioni. Un'altra comodità dei sistemi di comunicazione odierni è la comunicazione senza errori, anche se lo strato sottostante è caratterizzato da perdite di pacchetti. Questo viene realizzato da un sistema di scoperta e ritrasmissione dei messaggi persi, ma il tempo richiesto per scoprire la perdita del messaggio, la ritrasmissione e la ricezione introduce una latenza importante. È possibile cercare di minimizzare la perdita di pacchetti (con conseguente ciclo di ritrasmissione) applicando ad esempio un aumento della potenza di trasmissione in caso di alte perdite.
- Riordinamento. I messaggi possono arrivare in ordine diverso da quello di spedizione. Un esempio è dato dai multiple path, attraverso cui un pacchetto spedito più tardi ma su un path più veloce arriva a destinazione prima di uno che viaggia su un path lento ma che è stato spedito prima. Il messaggio che arriva prima può essere messo ad attendere il tempo idoneo oppure può essere consegnato subito. Il sistema di riordinamento pacchetti, tipico del TCP, può introdurre una importante latenza, che può essere ridotta sostanzialmente in due modi: o cercando di migliorare il sistema di riordinamento (non usando TCP che non permette un riordinamento intelligente dei pacchetti) o eliminandolo proprio, sapendo però che i pacchetti arriveranno non ordinati. Va ricordato però che se un pacchetto da riordinare è stato perso il successivo riordinamento richiede maggior tempo dato che implica anche una ritrasmissione. Il TCP che deve garantire una buona connessione implementa un sistema di ritrasmissione esponenziale che può portare i tempi di ritrasmissione anche a minuti. Si può supporre che un nuovo protocollo possa implementare un sistema migliore di ritrasmissione oppure non lo implementi affatto, dato che potrebbe non servire (se ricevo pacchetti in tempo reale con informazioni della velocità istantanea perderne uno ma ricevere il precedente e il seguente non cambia l'effetto macroscopico di regolazione della velocità del mezzo).

- Batching. Per riuscire ad abbassare la latenza bisogna ottimizzare ed usare in modo intelligente le risorse a disposizione. Alcuni degli esempi più significativi al riguardo sono la gestione ottimale della dimensione degli header dei pacchetti da inviare, che influisce sul numero di interrupt generati dal sistema nell'unità di tempo, dato che per ognuno di loro il processore deve decodificare il pacchetto e 'scartare' il payload. Oltre a ciò la dimensione dell'header influisce anche sulla occupazione della banda interessata alla trasmissione. Se il payload di un pacchetto è piccolo rispetto al MTU si possono addizionare più pacchetti e inviarli insieme; questo, pur confliggendo con la politica di mandare un singolo pacchetto per messaggio, può però sicuramente migliorare l'efficienza della comunicazione. Un'altra caratteristica di alcune schede odierne è l'invio di un'interrupt alla CPU non ogni volta che arriva un singolo pacchetto alla scheda ma solo dopo aver ricevuto più pacchetti; questo alleggerisce il numero di interrupt che vanno serviti. Diminuisce così la latenza data dal servizio dell'interrupt, ma aumenta la latenza di consegna del singolo messaggio. Si sceglie quindi di favorire l'efficienza del sistema a discapito della latenza del messaggio. Questo caratteristica può essere 'trasparente' se la frequenza di arrivo dei messaggi è alta. Il problema si presenta invece quando non si conosce a priori il tempo di inter-arrivo dei pacchetti. Inoltre si può considerare di richiedere un minimo di dati per dare il via ad una trasmissione (un esempio è l'algoritmo di Nagle per ritardare l'invio dei pacchetti nel TCP e migliorare l'efficienza). Per quanto riguarda un sistema a bassa latenza questa implementazione è sconveniente, è meglio mettere a disposizione un protocollo che assicura una latenza minima e lasciare all'applicazione la gestione dell'invio e della ricezione dei pacchetti stessi.
- Scheduling della CPU. Un punto critico di un sistema di comunicazione è il tempo del processore (CPU TIME). C'è sempre un po' di ritardo tra quando un messaggio è pronto e quando viene schedulato per l'invio. Generalmente la latenza dello scheduler è composta da tempi fissi e variabili. Ad esempio il tempo da quando viene ricevuto in interrupt a quando la CPU lo serve quando quest'ultima si trova in idle è fisso, mentre è variabile a seconda se ci sono o no CPU libere e a quali risorse deve essa accedere per servire quell'interrupt (context switching e/o I/O lock)
- Buffer delle socket. I Buffer delle socket sono i buffer di invio e ricezione presenti nelle socket del kernel. Questi buffer aiutano a migliorare la connessione, smussando ad esempio i picchi in arrivo o in uscita; d'altro canto come in ogni buffer viene introdotto un ritardo al sistema di messaggistica. È una latenza molto variabile, da zero ad un ipotetico risultato

peggiore che si ottiene dividendo la dimensione del buffer per la frequenza di passaggio dei dati. La cosa migliore è sempre cercare di farne decidere l'uso all'applicazione (che può disabilitarli per una minore latenza a discapito di una perdita di efficienza).

- Code all'interno della rete. Switches, router, access point wireless hanno di solito dei buffer in cui vengono messi in coda messaggi interi o anche frazioni di essi, prima di inoltrarli verso la destinazione. Come sempre l'obiettivo di questi buffer è lo smussamento dei picchi nella comunicazione. Sistemi desktop o consumer tendono ad avere un buffer ridotto, mentre un router per centri dati o per una dorsale di rete può bufferizzare centinaia o migliaia di pacchetti per porta. Ogni pacchetto bufferizzato introduce una certa latenza dovuta al passaggio nel buffer. La latenza all'interno della rete è presente tutte le volte che c'è un qualche sistema incaricato di inoltrare pacchetti. Questa latenza risente sia del traffico presente sulla rete che del tipo di sistema di forwarding adottato. La latenza può essere minima quando il sistema di forwarding deve unicamente decidere su quale porta ridirezionare il pacchetto in una rete a basso traffico, fino ad un tempo molto lungo in una rete congestionata. Questo aspetto è sicuramente importante in una rete molto grande e complessa mentre l'impatto è minore in una rete con poche stazioni.
- Latenza di Serializzazione. Esistono diverse tecniche di serializzazione dei dati per spostarli attraverso la rete. Tipicamente viene utilizzata una trasmissione sincrona al tempo di clock, un bit ogni colpo di clock. La serializzazione introduce latenza, data dal fatto che un ricevitore non può usare il dato finché non e' arrivato l'ultimo bit di informazione del pacchetto. La comune linea telefonica analogica, che opera attorno ai 56Kbps, aggiunge circa 214ms di latenza di serializzazione ad un pacchetto di 1500 Byte. Una linea ADSL operante attorno ai 1.5Mbps (2.0 nominali), aggiunge 8ms di latenza. Anche un collegamento ethernet a 100Mbps aggiunge una latenza di 120 μs (sempre su un pacchetto di 1500 byte). La latenza di serializzazione è costante e dipende dal clock rate, ed è quindi la stessa per ogni pacchetto trasmesso, come nel caso di una rete wireless.
- Ritardo di propagazione. Sappiamo che 300.000 km/s è la velocità della luce e un'onda radio si propaga in aria a circa la stessa velocità con condizioni atmosferiche favorevoli. Sulle corte distanze il tempo di propagazione si può considerare costante e praticamente insignificante mentre il ritardo di propagazione diventa importante per comunicazioni su cavo, dato che la velocità è circa il 60% di quella della luce nel vuoto. Ad esempio per una trasmissione su 3000km di cavo verrebbero impiegati 15ms. In ogni caso i ritardi di

propagazione vanno considerati costanti per una trasmissione su cavo, in aria su lunghe distanze potrebbero variare significativamente con le condizioni atmosferiche.

6.10 Latenza e concetto di priorità

Dalla analisi di alcune delle principali cause di latenza effettuata emerge chiaramente come un utile contributo al problema di minimizzarla possa venire dal concetto di priorità. Ragionando in termini di sistema appare evidente che se è possibile soddisfare per prime le richieste più 'urgenti', il risultato che si otterrà sarà in generale una aumentata capacità di risposta del sistema entro prefissati termini temporali, e quindi una molto maggiore capacità di controllare la dinamica di un evento esterno.
Partendo da questo dato assolutamente generico vediamo come si possa intervenire in modo da massimizzare le prestazioni nel caso di nostro interesse, ad esempio introducendo gestione di priorità sui messaggi (o pacchetti) e sui task (o processi) all'interno del sistema operativo, oltre che nel protocollo progettato. Tali tecnologie e strumenti sono stati utilizzati nel corso del presente lavoro e verranno nel seguito introdotti e discussi.

6.10.1 Latenza e Sistemi Operativi: Linux Real-Time

Come già più volte ricordato nel corso del presente lavoro è stata fatta la scelta di utilizzare esclusivamente software libero, e concordemente in questo paragrafo si considereranno tra i sistemi operativi real-time solo quelli appartenenti alla famiglia 'Linux'. Sul mercato esistono in realtà molte alternative possibili, ma nessuna presenta i rilevanti vantaggi di economicità, disponibilità e modificabilità dei sorgenti, e quindi personalizzabilità offerte dalle soluzioni aperte. Tali vantaggi sono stati ritenuti tanto significativi per lo sviluppo del nuovo protocollo Standard che altre possibilità non sono semplicemente state considerate. Ci concentreremo quindi nel seguito sul mondo dei sistemi operativi Real-Time Linux based.
Brevemente ricordiamo una definizione di sistema Real-Time tra le tante disponibili.
Un sistema real-time è un sistema nel quale la correttezza delle elaborazioni non dipende solo dalla loro esattezza logica ma anche dall'istante in cui i risultati vengono prodotti. Se in vincoli temporali associati al sistema non vengono rispettati, allora il sistema è comunque in uno stato di errore.
In altre parole il sistema deve essere deterministico e garantire un certo comportamento rispetto ai tempi di output anche al variare del carico. E' importante notare che la definizione

appena introdotta non parla di prestazioni velocistiche, ed infatti il real-time non è una questione di velocità: è una questione di predicibilità.

Ad esempio usando un processore aggiornato è facile ottenere tempi di risposta di pochi microsecondi, ma occasionalmente in un sistema per utilizzo generale si hanno tempi di risposta enormemente più lunghi, anche dell'ordine di un secondo. E proprio qui è il problema fondamentale: il sistema non è che non sia sufficientemente veloce o efficiente, però il suo comportamento non è deterministico, ovvero non predicibile.

Nel caso del presente lavoro è utile ottenere la massima predicibilità possibile per task che svolgono funzioni real-time sulla macchina, ad esempio gestione di dati per il controllo distribuito, e preparazione, invio e ricezione di messaggi che consentano di realizzare efficacemente tale controllo. Per ottenere questo risultato e contemporaneamente consentire l'esecuzione di ulteriori task dedicati ad altre applicazioni o alla gestione del sistema è chiaramente indispensabile un sistema di priorità assegnabili articolato, oltre ad una lunga serie di altri accorgimenti a cui verrà nel seguito fatto cenno. Nella Figura 6.5 a titolo di esempio sono mostrati i livelli di priorità associati ad un kernel real-time in rapporto ai livelli tipici di un kernel mainstream.

Dalla figura è immediatamente evidente che i livelli del kernel real-time sono un superset di quelli 'normali'. Ciò è molto utile in quanto si può fare in modo che all'interno del sistema una applicazione real-time giri con priorità maggiore delle applicazioni normali, senza che ci sia bisogno di alcuna modifica ne ricompilazione di queste ultime.

Applicando un concetto simile anche alla gestione del traffico possiamo ottenere sistemi che conciliano un buon comportamento per quanto riguarda il controllo real-time anche distribuito con la possibilità di svolgere i 'normali' compiti associati a sistemi di elaborazione delle informazioni.

In realtà la possibilità di utilizzare Linux come sistema operativo per applicazioni embedded con tutti i requisiti tipicamente ad esse collegati, è da tempo perseguita dall'industria e dagli ambienti di ricerca.

Alcune delle ragioni di tale interesse sono quelle più volte citate, come ad esempio la mancanza di royalties da pagare e la disponibilità di librerie, tool di sviluppo e applicazioni per una amplissima casistica di situazioni. Però esistono anche motivi di natura strettamente tecnica, ad esempio l'abilità di gestire task real-time e 'best effort', ovvero di tipo tradizionale, sulla stessa macchina, come evidenziato in dettaglio qualche riga più sopra.

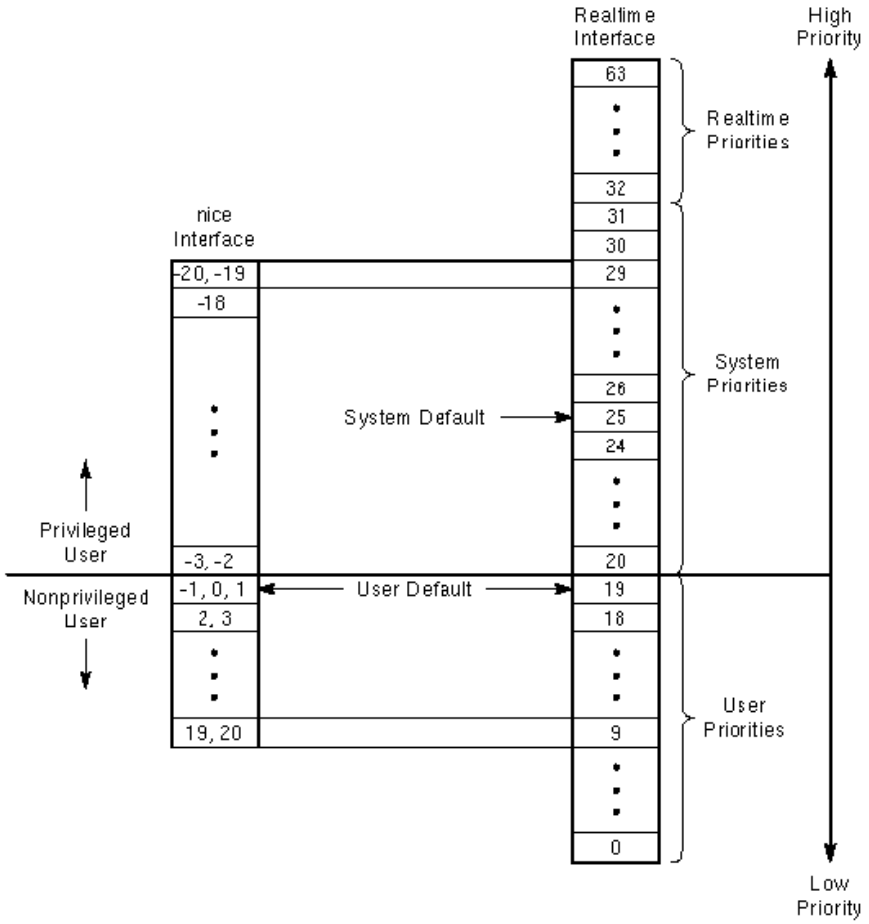


Figura 6.5: Livelli di priorità del kernel real-time

Come risultato molti vendor e anche università **[60]** stanno supportando attivamente Linux nel suo percorso di 'avvicinamento' al real-time, oppure sviluppando distribuzioni o interi ambienti real-time commerciali **[58]**, **[59]**, **[61]** partendo da una base Open Source.

Il metodo più diffuso sinora di aggiungere capacità real-time a Linux si potrebbe definire l'approccio 'doppio kernel', nel quale le applicazioni real-time sono gestite da un kernel dedicato che vede un altro kernel standard come uno dei suoi task. In pratica i due kernel sono come innestati uno nell'altro.

Più di recente ha iniziato ad emergere un approccio alternativo, ovvero di migliorare direttamente le capacità in termini di real-time del kernel standard di Linux, grazie agli sforzi del progetto 'RT Patch' **[62]**, **[63]**. Al momento quest'ultimo approccio è caratterizzato da prestazioni inferiori, che potrebbero non essere sufficienti per alcune classi di applicazioni di nicchia con stringenti specifiche. Però ha vantaggio di avere una struttura concettuale e implementativa più lineare e quindi semplice da implementare e mantenere, ed inoltre è già

ora di interesse per applicazioni con specifiche meno stringenti quali quella oggetto del presente documento, che non richiede un real-time 'hard', ma piuttosto di tipo 'soft'.
I vantaggi citati potrebbero diventare molto rilevanti con il procedere dello sviluppo, ed in particolare visto che una parte della RT Patch è in corso di integrazione con il kernel standard, e quindi almeno una parte delle sue funzionalità saranno disponibili in tute le distribuzioni. Inoltre ciò probabilmente imprimerà una accelerazione nello velocità e qualità dello sviluppo, a tutto vantaggio delle prestazioni ottenibili in futuro. Oltre a ciò la 'RT Patch' già ora offre il grande vantaggio di essere stata integrata in una importante distribuzione commerciale **[61]**, e nei suoi derivati free, aprendo la possibilità di sfruttare contemporaneamente sia buone prestazioni in termini di real-time che nell'immediato uno sterminato parco software applicativo e di sviluppo, sia libero che commerciale.
In effetti nel corso del presente lavoro proprio quest'ultima possibilità è stata scelta è utilizzata estensivamente, e viene quindi brevemente introdotta nel seguente paragrafo.

6.10.2 Red Hat Enterprise MRG e derivati

Le versioni di Linux utilizzate nel corso del presente lavoro sono in realtà un derivato 'Open' della distribuzione RedHat versione Enterprise MRG, ottenute ricompilando la stessa a partire dai sorgenti, ovvero CentOS **[64]** frutto di un progetto dedicato e Scientific Linux **[65]** messa insieme dal Fermilab del CERN di Ginevra e vari altri laboratori e università in tutto il mondo. Poiché però il progetto originario della 'RT Patch' e la sua iniziale implementazione sono strettamente collegate alla distribuzione madre, e vi è in pratica quasi totale uniformità tra questa e le derivate, si è ritenuto più corretto nel presente lavoro citare la distribuzione di riferimento, ed elencarne brevemente le caratteristiche più importanti.
Red Hat Enterprise MRG è in realtà un ambiente IT complesso di nuova e recente generazione dotato di funzionalità di Messaging, Grid computing e Realtime, quindi decisamente variegate. Poiché di interesse per il presente lavoro sono sopratutto le caratteristiche real-time di seguito è inserita una breve presentazione che si riferisce unicamente a queste. Per ulteriori approfondimenti si rinvia ai riferimenti citati.
La implementazione di MRG Realtime in realtà consiste in un lungo elenco di caratteristiche e modifiche il cui scopo finale è assicurarsi che task riferiti ad applicazioni con priorità real-time 'girino' nel sistema operativo con la massima priorità possibile e siano interrotte il meno possibile per la gestione di processi a priorità più bassa e servizi di sistema.

Di seguito è presente un elenco dei miglioramenti tecnici fondamentali che sono stati introdotti in MRG Realtime:

- Prestazioni fortemente deterministiche: MRG Realtime incrementa drammaticamente il determinismo di Linux. Sostituendo il kernel standard con uno dedicato, MRG Realtime incrementa significativamente le prestazioni per quanto riguarda lo scheduling e fornisce full preemption per raggiungere tempi di intervento che arrivano anche sotto la decina di microsecondi.
- Timer ad alta risoluzione: MRG Realtime include il supporto per timer con una accuratezza al nanosecondo. Inoltre molte funzioni sono state ottimizzate. Ad esempio la funzione gettimeofday() utilizzata da molte applicazioni per timestamping nei log files è stata riscritta in modo da non richiedere un context switch, in modo da ottenere prestazioni notevolmente migliorate.
- Piena compatibilità delle applicazioni con la Red Hat Enterprise Linux standard: MRG Realtime non apporta nessuna modifica in userspace. Per questo motivo la piena compatibilità con applicazioni dell'ambiente Red Hat enterprise e derivati è mantenuta. Ciò significa che qualsiasi applicazione per Red Hat linux e Linux in generale gira anche su MRG Realtime senza necessità di modifiche ne di ricompilazione.
- Gestione degli interrupt a bassa latenza: i lunghi e non interruttibili percorsi di esecuzione del codice contenuti tipicamente nei gestori di interrupt sono una delle cause primarie di mancanza di determinismo nei sistemi. MRG Realtime risolve il problema spezzando i lunghi percorsi di esecuzione in piccole porzioni separatamente schedulabili. Ciò assicura che la gestione di eventi a più bassa priorità non blocca l'esecuzione di task real-time.
- Gestione migliorata delle priorità: MRG Realtime include una regolazione dei parametri che prevede maggiore granularità. Per esempio è possibile associare al sottosistema di rete una priorità maggiore rispetto alla gestione dei dischi e del restante storage. Oltre a ciò sono stati implementati miglioramenti dello scheduler che garantiscono che processi a priorità maggiore siano eseguiti senza interruzioni.
- Verifica della non inversione delle priorità: MRG Realtime, usata insieme alla libreria glibc, fornisce ereditarietà delle priorità. Tale meccanismo serve ad essere sicuri che processi a priorità più bassa non blocchino altri a priorità maggiore che si contendono le stesse risorse.
- Precisione maggiorata dei timer: tute le applicazioni realtime richiedono eventi temporizzati in maniera molto accurata. Ciò serve ad esempio per avere messaggi con timestamp precisi,

oppure timeout che scadano esattamente quando previsto. Linux tradizionalmente è dotato di un sistema di conteggio del tempo la cui accuratezza è affidata ad un interrupt periodico, che però fornisce risultati tutt'altro che precisi. Al contrario MRG Realtime usa timer basati su eventi hardware, ottenendo come risultato una precisione sostanzialmente migliorata.

- Sensore di latenza: il sensore di latenza è un rivelatore di picco. E' in grado di identificare i percorsi di esecuzione a maggiore durata e non interrompibili del kernel, ed è uno strumento molto utile per comprendere se la causa di risposte non sufficientemente deterministiche è nel kernel piuttosto che nella applicazione userspace che si sta sviluppando.
- Tuna: E' uno strumento utile per configurare il sistema. Infatti un sistema su cui gira MRG Realtime deve essere opportunamente regolato per offrire le migliori prestazioni possibili. Tuna consente di farlo 'on the fly' tramite interfaccia grafica mentre il sistema è sotto carico, semplificando molto l'ottenimento della configurazione ottimale.

6.10.3 Latenza e gestione del traffico di rete

Un notevole contributo alla riduzione della latenza, reso ancora più significativo in quanto legato al sottosistema di gestione della rete che è lo specifico settore di interesse del presente progetto, può venire dalla implementazione di una gestione efficace del traffico. Come già accennato in precedenza il nuovo protocollo si vorrebbe venisse utilizzato in un contesto nel quale fosse presente traffico a vari livelli di priorità legato alle comunicazioni di pertinenza del protocollo stesso, ma anche altro traffico legato alle normali attività di sistema. E' di tutta evidenza che il sistema a cui si fa riferimento nel presente lavoro è tipicamente un controller a bordo di un mezzo agricolo, e pertanto il tipo di traffico di interesse non sarà quello di un normale sistema PC desktop, ma è pur vero che si possono immaginare situazioni in cui anche a bordo di un mezzo agricolo possono essere utili informazioni ulteriori oltre alla fondamentale sincronizzazione real-time. Ad esempio dati per il fleet management, o per la gestione della produzione, o comunicazioni tra gli operatori a bordo dei veicoli e una base operativa nella azienda agricola. Il traffico generato per svolgere queste ulteriori funzioni avrebbe sicuramente una priorità inferiore, ma verosimilmente per la sua trasmissione verrebbero utilizzate le stesse risorse di calcolo e trasmissive. La situazione ottimale si otterrebbe nel momento in cui si riuscisse ad imporre al sistema nel suo complesso una politica di gestione del traffico dettata direttamente dal nuovo protocollo Standard che si intende realizzare, e quindi che venissero rispettate alcune classi di priorità ben definite e

considerato automaticamente il traffico non classificato come quello a priorità inferiore. In questo modo si otterrebbe il desiderabile risultato di mantenere buone prestazioni per quanto riguarda il real-time mantenendo però la compatibilità con altre applicazioni, che anzi andrebbero sviluppate e utilizzate senza cambiamenti.
Questo obiettivo è raggiungibile tramite una combinazione di passi, che comprendono dalla configurazione del sistema, all'utilizzo di componenti aggiuntivi, alla particolare etichettatura del traffico. Nel seguito saranno descritti alcuni dei concetti fondamentali, partendo dal comportamento di default del kernel Linux.

6.10.4 Standard Linux kernel Traffic Management

Linux offre un ricco set di strumenti per gestire e manipolare la trasmissione di pacchetti. Esistono letteralmente centinaia di tool e servizi di rete che possono girare su tale sistema operativo che permettono di implementare politiche di gestione del traffico anche decisamente sofisticate. Oltre a ciò anche il sottosistema di gestione della rete implementato nel kernel ha una struttura articolata e sofisticata. A titolo di esempio in Appendice C per consentire una migliore leggibilità, Figura C.1, è riportato il diagramma completo che descrive tutti i blocchi di elaborazione presenti nel sottosistema di rete del kernel Linux attuale, versione 2.6.
Controllo del Traffico (ovvero Traffic Control, nel seguito TC) è il nome attribuito ad un sistema di code e più in generale di meccanismi tramite i quali i pacchetti che formano il traffico di una rete TCP/IP vengono ricevuti e trasmessi da un nodo della rete stessa. Le operazioni normalmente eseguite comprendono ad esempio decidere quali e quanti pacchetti accettare e a che ritmo all'ingresso di una interfaccia di rete, ed anche quali pacchetti, in che ordine e a che ritmo trasmettere pacchetti in uscita da una interfaccia.
La situazione più semplice che si può immaginare di TC, che corrisponde a quanto accade in molte situazioni, consiste di una singola coda che raccoglie tutti i pacchetti in ingresso e in seguito li ritrasmette alla massima velocità alla quale l'hardware seguente o il sistema operativo sono in grado di gestirli.
Un coda di questo tipo è detta in inglese First In First Out, ovvero FIFO.
La politica di default di TC di Linux in ingresso ha un comportamento di questo tipo, mentre in uscita è implementata una politica un pò più sofisticata, che è caratterizzata da un numero maggiore di code. Avere una singola coda per tutto il traffico può comportare problemi ad esempio nel momento in cui il traffico generato da due macchine concorrenti dovesse saturare

il canale trasmissivo. Se ammettiamo che una delle due macchine possa generare la quasi totalità del traffico in maniera tale da saturare le code disponibili, l'altra macchina potrebbe non riuscire a comunicare in rete vedendosi scartare gran parte dei pacchetti a causa del riempimento dell'unica coda disponibile.

La separazione delle code associate a tipi di traffico diversi in un caso del genere può essere molto utile per far sì che diverse applicazioni possano condividere in modo efficace lo stesso canale trasmissivo. TC è il set di strumenti che consente di avere un controllo granulare sulle code legate alle interfacce di rete e sui loro meccanismi di gestione, e può essere molto utile ad implementare la politica di gestione a priorità richiesta dal nuovo protocollo.

Nel nostro caso si vorrebbe ottenere bassa latenza per le applicazioni che richiedono prestazioni real-time, consentendo però anche la contemporanea trasmissione di dati a priorità normale.

Nella seguente Figura 6.6 è evidenziata la struttura di gestione del traffico utilizzata di default in uscita nel kernel linux 2.6:

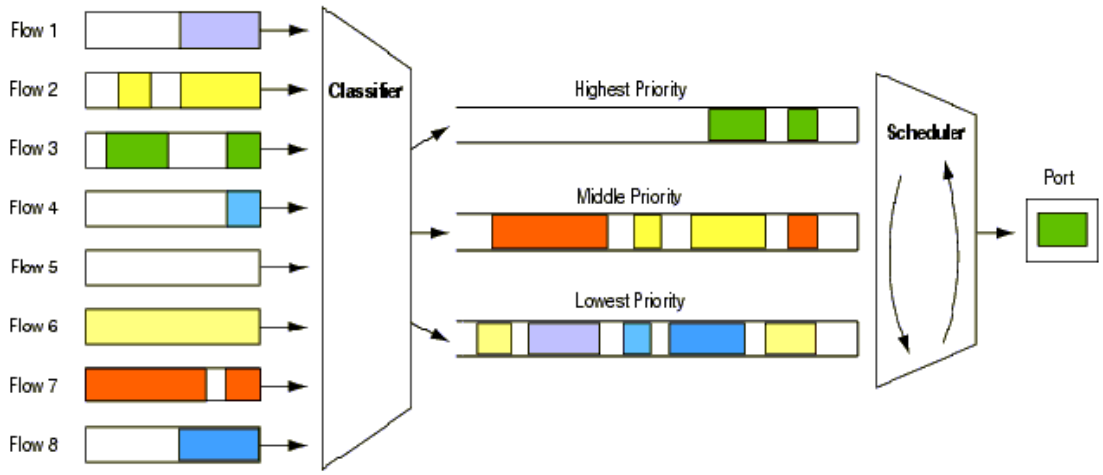


Figura 6.6: Default output policy del Linux kernel 2.6

Il nome che nel kernel Linux viene dato a strutture di questo tipo è qdisc. Un qdisc detto semplicemente è una sorta di scheduler associato ad una interfaccia di rete. In realtà un qdisc è qualcosa di più complesso, dato dalla associazione di alcune strutture dati alle quali è possibile associare dei comportamenti, ottenendo come risultato delle politiche anche decisamente complesse. Però per quanto riguarda il presente lavoro la definizione introdotta è sufficiente, e si è ritenuto inutile approfondire troppo la questione. In ogni caso ulteriori riferimenti per reperire ulteriori elementi sono segnalati a fine paragrafo. La struttura evidenziata in figura fa

riferimento al qdisc detto ‘pfifo_fast’, ed è in pratica ottenuta dalla combinazione di tre code FIFO, che singolarmente non effettuano nessun speciale trattamento sui pacchetti. Quello che però realizza una gestione del traffico è la suddivisione stessa del traffico in code separate e la politica di svuotamento delle code stesse. Finché sono presenti pacchetti in una coda a più alta priorità le altre code non vengono processate. Il kernel Linux onora il flag ‘Type of Service’ presente nei pacchetti IP, ed automaticamente inserisce i pacchetti contrassegnati a minimo ritardo (‘minimum delay’) nella coda a massima priorità.

Questo comportamento è molto simile a quanto si vorrebbe per il progetto del nuovo protocollo nel campo agricolo oggetto del Task 2, ed è stato estensivamente utilizzato nei test preliminari. Il protocollo si propone di estendere ad un numero maggiore di livelli di priorità la gestione del traffico, ma molto probabilmente ciò si può ottenere tramite una semplice ‘riprogrammazione’ del kernel e opportuna marcatura del traffico in modo da implementare il comportamento richiesto. Ulteriori opzioni sono state considerate, viste le varie possibilità di sviluppo del nuovo Standard, ma il loro trattamento esula dagli scopi del presente documento.

E’ molto importante notare però che mentre la politica di default per il traffico in uscita da un sistema Linux è già molto vicina al caso ideale, così non avviene per quanto riguarda l’ingresso. Infatti il TC sotto Linux non consente di implementare in ingresso politiche di gestione del traffico altrettanto sofisticate, anzi l’unica disponibile è del tipo singola coda FIFO, con la possibilità aggiuntiva di poter eventualmente solo scartare dei pacchetti in ingresso. Ciò non è sufficiente per gli scopi del presente lavoro, che richiederebbero una gestione della priorità anche per i pacchetti in ingresso. Nel seguito dunque verranno introdotti due strumenti che consentono di ottenere questo risultato, dei quali il primo è immediatamente disponibile, mentre il secondo, il cui sviluppo è stato avviato nel corso del presente Dottorato ed è attualmente in fase di test, sarà disponibile a breve. Per ulteriori approfondimenti sulle sofisticate possibilità offerte dal Controllo del Traffico sotto Linux sono presenti molti documenti in rete ed alcuni riferimenti in bibliografia **[66]**, **[67]**.

6.10.5 Linux InterMediate Queuing device (IMQ)

Linux InterMediate Queuing device (nel seguito IMQ) è una ‘patch’ o modifica che si può applicare al kernel Linux standard, e richiede una riconfigurazione e ricompilazione del kernel stesso per poter essere utilizzata.

IMQ ha due utilizzi principali, il primo è di applicare all'ingresso di una interfaccia di rete delle politiche sofisticate di gestione del traffico, cosa che non è consentita dal kernel standard. Infatti agli ingressi è possibile associare dei qdisc, ma la loro funzionalità è decisamente limitata. Di converso esistono qdisc associabili alle uscite che sono estremamente flessibili ed efficienti, e possono realizzare politiche decisamente sofisticate. Con IMQ è possibile utilizzare i qdisc di uscita per modificare le politiche in ingresso, in pratica utilizzandoli come se fossero applicati agli ingressi.

L'altro utilizzo tipico di IMQ è di implementare politiche di gestione traffico su più interfacce contemporaneamente ed in maniera correlata. Normalmente è possibile collegare un qdisc ad una sola interfaccia. E' possibile collegare lo stesso tipo di qdisc a quante interfacce si vuole, ma ogni istanza non è in relazione con le altre. In pratica che non viene condiviso lo stato, nè vi può essere scambio di informazioni tra queste. Quindi ad esempio normalmente non è possibile limitare la banda in modo che la somma dei dati che scorrono attraverso due distinte interfacce di rete sia sotto un certo limite. In una situazione del genere è utile IMQ, che risolve il problema.

Nel caso del presente Dottorato la prima funzionalità è molto rilevante, in quanto permette di implementare una politica di priorità nella gestione dei pacchetti anche in ingresso, cosa che non è possibile con il kernel Linux standard.

Nella seguente Figura 6.7 è mostrato uno schema che descrive il comportamento in ingresso del kernel Linux senza modifiche:

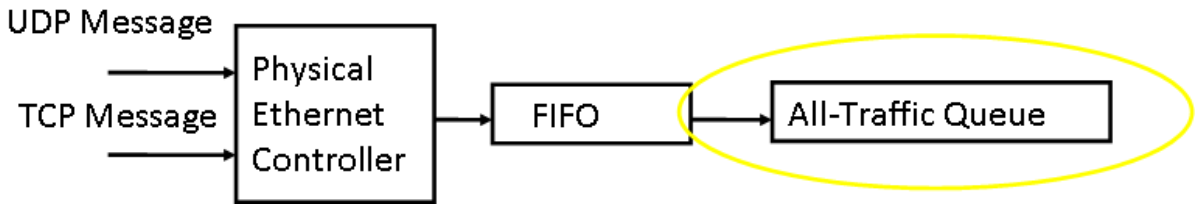


Figura 6.7: Default input policy del Linux kernel 2.6

Nella realtà la situazione è più complessa, ma la figura rende bene l'idea. Sono disponibili solo buffer che realizzano code FIFO messe una di seguito all'altra, e l'unica azione consentita è lo scartare alcuni pacchetti selezionandoli dal flusso complessivo. Dunque manca la possibilità di implementare una politica di priorità né è possibile 'ritardare' un tipo di traffico rispetto ad un altro.

Per modificare tale comportamento IMQ applica una strategia come quella mostrata in Figura 6.8:

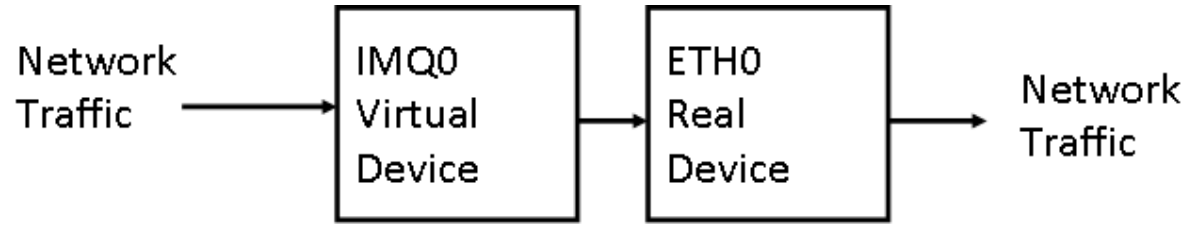


Figura 6.8: Linux IMQ concept

In pratica la patch crea una nuova interfaccia di rete 'virtuale' che si può interporre alle interfacce reali, o anche collegare in modi più complessi, da cui il nome InterMediate Queuing device.

Così facendo la situazione finale risultante è mostrata in Figura 6.9:

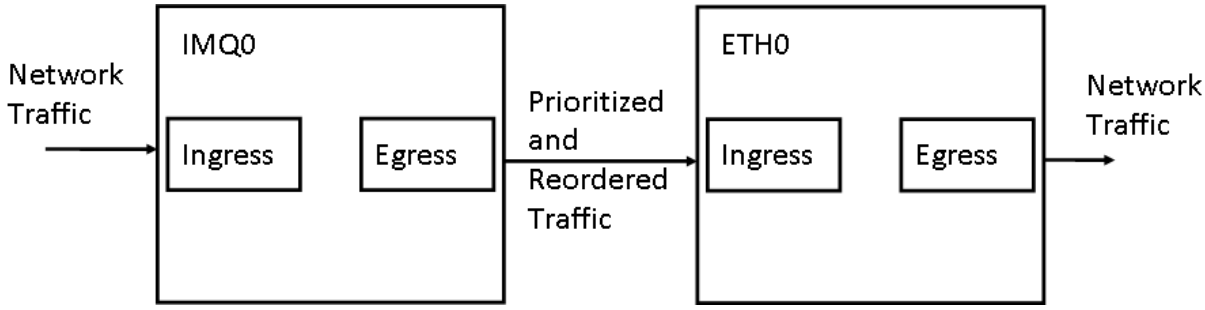


Figura 6.9: Linux IMQ - Ingress ed Egress

Il vantaggio consiste nel fatto che in questo caso l'uscita della interfaccia virtuale IMQ0 diventerà dal punto di vista delle politiche sul traffico equivalente all'ingresso della interfaccia fisica EH0 ad esempio.

In altre parole potrò utilizzare il qdisc di uscita di IMQ0 per impostare politiche sul traffico che globalmente passa nella 'macro-interfaccia' di rete formata da IMQ0+ETH0, ottenendo quindi il desiderato risultato di poter implementare una politica di priorità.

In astratto questo approccio parrebbe risolvere tutti i problemi e soddisfare le richieste associate alla implementazione del nuovo protocollo oggetto del presente Dottorato. Purtroppo però anche questo approccio presenta alcuni lati negativi, che hanno portato ad avviare lo sviluppo dello strumento di cui si parlerà nel prossimo paragrafo.

Il primo problema che era legato probabilmente ad una certa immaturità del codice di IMQ, era un avvertibile calo prestazionale, ma pare superato nelle ultime release, in particolare a partire dalla patch per il kernel 2.6.34.
Oltre a ciò rimane una certa complessità e mancanza di linearità architetturale, in quanto il traffico è comunque costretto ad attraversare due interfacce di rete e quindi tutte le strutture dati ad esse associate nel kernel. Questo impostazione un pò 'sbrigativa' nella architettura di Linux IMQ è probabilmente anche il motivo per cui sinora questa interessante funzionalità non è stata ancora inglobata nel kernel mainstream.
In ogni caso le prestazioni ottenute da test recenti sembrano interessanti, e nel caso di ulteriori evoluzioni, e/o affinamenti architetturali potrebbe rivelarsi una scelta interessante per la sua estrema flessibilità. In bibliografia sono presenti riferimenti atti al reperimento di ulteriori informazioni **[68]**, **[69]**.

6.10.6 Agro Kernel

A causa delle criticità riscontrate durante l'analisi e l'utilizzo di Linux IMQ durante il corso del presente dottorato è stato avviato lo sviluppo di una nuova patch del kernel, chiamata 'Agro Kernel' con riferimento al settore agricolo nel quale si pensa di utilizzarla.
In realtà la patch si propone di risolvere un problema più generale, che è stato probabilmente trascurato dal gruppo che ha originariamente implementato il Controllo del Traffico sotto Linux (TC nel seguito).
Infatti nella documentazione riguardante Linux TC ricorre il concetto che non ha senso implementare politiche di gestione del traffico in ingresso in quanto chi riceve non ha la possibilità di influenzare il traffico spedito. Questa argomentazione è assolutamente vera, e nel caso si consideri TC solo come uno strumento per limitare la banda, e si faccia riferimento al solo protocollo TCP, è anche probabilmente vero che in ingresso ha senso solo la possibilità di scartare dei pacchetti, che associata al tipico comportamento di TCP ottiene come effetto collaterale visto dal receiver il risultato di limitare la banda in uscita del sender.
Però può accadere, ed infatti accade proprio nel caso del presente lavoro, che sia necessario impostare politiche che non siano riferibili alla semplice limitazione di banda. Ad esempio come già ampiamente descritto nel caso di un protocollo per la sincronizzazione realtime si vorrebbe la possibilità di privilegiare il traffico associato a compiti real-time ed alta priorità in

generale a discapito del traffico con meno stringenti esigenze. Per fare ciò è necessaria almeno una struttura di ricezione dei pacchetti in arrivo con code multiple.

La patch 'Agro kernel' si propone di realizzare una modifica concettualmente molto semplice ed elegante, ovvero consentire di implementare anche in ingresso gli stessi meccanismi disponibili in uscita, o almeno un sottoinsieme di questi. Con riferimento al 'Linux kernel packet traveling diagram' in Appendice C, Figura C.1 la modifica effettuata è a livello del blocco 'QOS Ingress'. In Figura 6.10 è evidenziato il risultato che si è voluto ottenere:

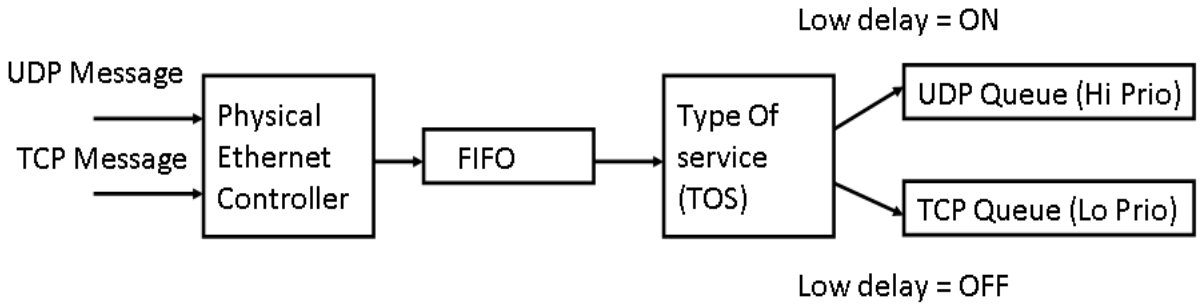


Figura 6.10: Agro patch traffic priority

Ovvero la implementazione di due code separate in ingresso al sottosistema di rete del kernel e subito successive all'hardware di rete, che distinguessero tra traffico UDP associato al protocollo e ad alta priorità dal restante traffico su TCP, implementando un meccanismo simile a quanto presente di default in uscita.

La versione del kernel utilizzata per implementare la 'Agro' patch è la 2.6.24.7, per motivi di compatibilità con altre patch (real-time e IMQ).

Pur essendo molto semplice dal punto di vista teorico però la realizzazione di tale obiettivo si è scontrata con la notevole difficoltà di programmare codice per il kernel Linux e di orientarsi nella complessa architettura software esistente, composta ormai da milioni di righe. Lavorare a livello kernel infatti richiede strumenti e metodi diversi dalla normale programmazione in userspace, ad esempio non si hanno più a disposizione le normali chiamate a funzioni di libreria, ma bisogna interfacciarsi direttamente alle funzionalità offerte dal kernel. Ancora le modifiche al kernel sono critiche per il debug, che risulta decisamente più complesso, e per produrre del codice che implementi un modulo, è necessario utilizzare alcune modalità particolari per riuscire a 'dialogare' con la restante porzione del kernel. Per dare una idea della complessità del codice del kernel in Figura 6.11 è mostrata una minima porzione del

diagramma funzionale che rappresenta la situazione all'interno del kernel 2.4 in prossimità della coda di ingresso al sottosistema di rete:

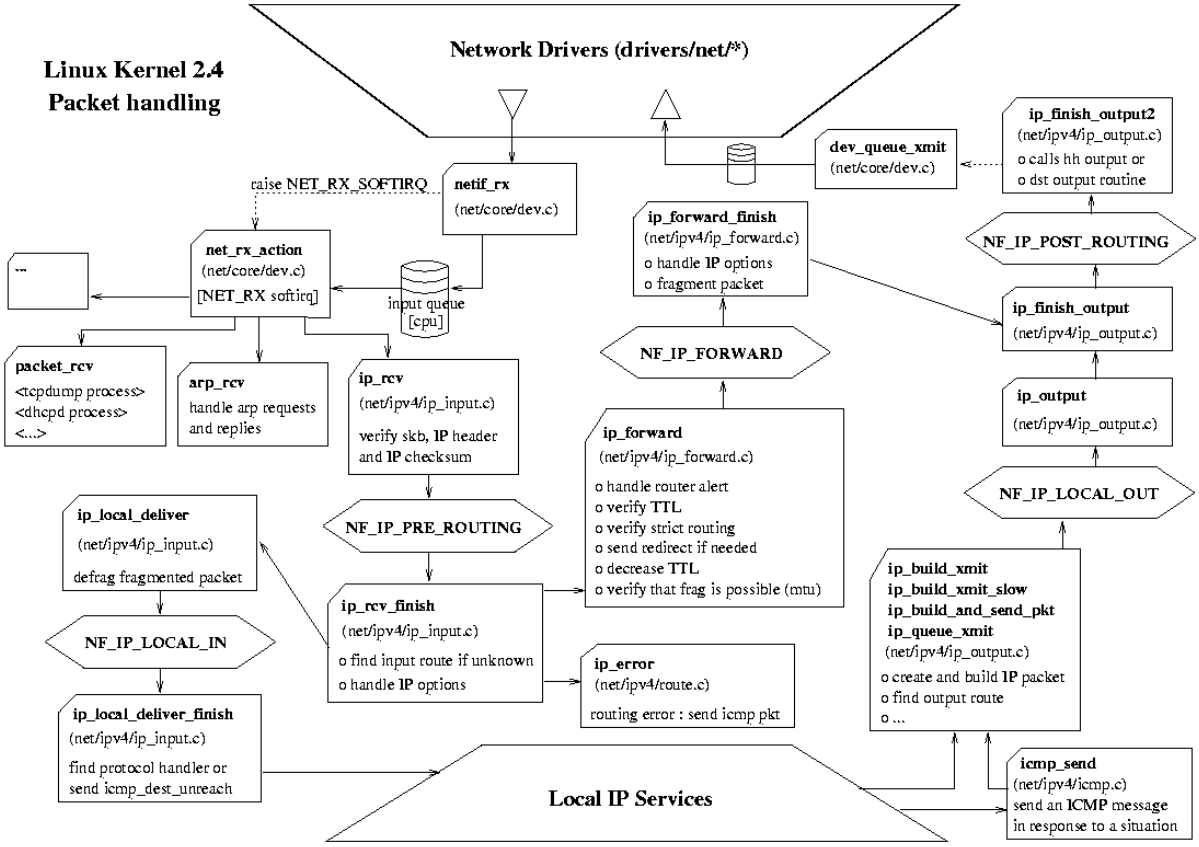


Figura 6.11: Diagramma funzionale di una porzione del kernel Linux 2.4

Basti dire che la complessità del kernel 2.6 è notevolmente superiore, visto che le funzionalità di rete dal kernel 2.4 sono state notevolissimamente ampliate (ad esempio supporto IPv6).
In ogni caso la patch è funzionante ed attualmente in fase di test, e il suo sviluppo potrebbe essere completato in pochi mesi. Sarebbe anche interessante iniziare il percorso di sottomissione della stessa patch al team di sviluppo del kernel mainstream, queste attività sono infatti inserite tra i possibili sviluppi futuri.

6.11 Task 2 - Hardware

L'approccio aperto utilizzato nella scelta del software utilizzato per il presente lavoro di dottorato è stato utilizzato anche nel caso dell'hardware. E' stato fatto dunque largo impiego di hardware commodity, ovvero normali sistemi disponibili sul mercato PC, oltre a sistemi in standard PC/104, che appartiene al mondo embedded.

Una scelta di questo tipo è sicuramente utile quando l'obiettivo è la creazione e lo sviluppo di un nuovo Standard Internazionale. Poiché però tali argomenti sono stati già trattati nel Capitolo 4 paragrafo 4.11.2 per quanto riguarda lo standard PC/104 e simili, in questa parte si introdurranno solo alcune ulteriori considerazioni sull'hardware di rete, sul quale in precedenza nulla è stato detto.

Per quanto riguarda quest'ultima categoria un esempio di hardware standard e di larghissimo consumo è quello che fa riferimento alla famiglia di tecnologie IEEE 802.11, che permette alte prestazioni con costi molto competitivi, oltre ad avere un vantaggio fondamentale per un nascente Standard Internazionale, ovvero la garanzia di frequenze assegnate in praticamente tutto il mondo. Se si operasse una scelta differente progettando un protocollo di rete ad-hoc sarebbe in effetti estremamente difficile consentire il suo utilizzo in differenti nazioni, visto che le frequenze disponibili non sono armonizzate. Questo vantaggio si rivela dunque fondamentale, ma non è l'unico.

Oltre a ciò lo standard 802.11 offre un livello di maturazione notevole, caratterizzato da implementazioni ormai molto performanti e con costi molto bassi.

Ad esempio con riferimento alla velocità di trasmissione, con le più recenti soluzioni compatibili con lo standard 802.11n questo parametro ha raggiunto il valore nominale di 300 Mbit al secondo, con valori reali misurabili che si approssimano alla metà di tale valore per brevi distanze, e ovviamente decadono all'incrementarsi della distanza stessa. Pianificare lo sviluppo di un hardware low cost con tali prestazioni sicuramente non è banale, e ancora meno semplice sarebbe reperire le necessarie frequenze libere in tutto il mondo.

Altro esempio è la tecnologia 'diversity' o MIMO, presente ormai anche in schede con costi di pochi euro; questa tecnologia consente di utilizzare più antenne invece di una per migliorare il segnale, e viene gestita direttamente in hardware. Funzionalità di questo tipo sono molto interessanti per applicazioni che riguardano stazioni mobili, dato che la qualità del segnale ricevuto da un mezzo in movimento è sicuramente molto incrementata dall'utilizzo di più antenne, opportunamente posizionate sul mezzo.

Ancora è possibile pensare alle tecnologie di cifratura native di IEEE 802.11 (ad esempio WPA2) che consentono un notevole livello di sicurezza a costi molto bassi senza aggiungere ritardi visto che sono anch'esse gestite dall'hardware.

Nel corso del presente lavoro si farà dunque riferimento ad hardware di rete aderente allo standard 802.11, introducendo ulteriori dettagli sulle specifiche soluzioni utilizzate nella parte sui test. Per quanto riguarda l'hardware di rete inizialmente anche altri protocolli sono stati

valutati, ad esempio Bluetooth o Zigbee, ma nessuno ha rivelato caratteristiche in termini di distanza raggiungibile, banda disponibile, affidabilità, competitività dei costi e sicurezza paragonabili alla famiglia 802.11.

6.12 Il Protocollo proposto

Nei successivi paragrafi saranno introdotte nozioni sul protocollo oggetto del Task 2, arrivando ad un certo livello di dettaglio nella sua descrizione. La gestazione del progetto è stata laboriosa in quanto come più volte ripetuto le esigenze da tenere in conto sono state numerose e svariate. Oltre a ciò è stato necessario sviluppare l'ambiente nel quale il nuovo protocollo si colloca, intendendo con ciò sia tutto il progetto della architettura tecnologica di contorno hardware e software, sia un ambiente di sviluppo vero e proprio, sia un parco macchine completo di software dedicati per i test. A questo proposito per dare un esempio della complessità ingegneristica delle soluzioni necessarie ad ottenere le prestazioni auspicate è stato inserito in Appendice C Figura C.2 uno schema preliminare in cui si delineavano alcune fasi successive della implementazione del protocollo sotto il profilo della architettura software. Oltre a ciò è anche da sottolineare che il protocollo stesso nasce per tentare di soddisfare stimoli proposti da ISO, che intendono rispondere a delle necessità definite ma non a tutti i livelli, nel senso che la normativa di riferimento è in parte ancora da costruire e in parte da completare, se non come indirizzo sicuramente nei dettagli implementativi. Ciò a significare che tutto l'insieme di soluzioni ideate e proposte è ancora in divenire, e dovrà in un prossimo futuro trovare compimento prendendo spunto dal confronto tra esigenze e possibilità tecniche e nuove evoluzioni della normativa. Tale dialogo dovrebbe portare nel giro di qualche anno alla definizione del nuovo Standard, è questo il senso e l'obiettivo che il presente Dottorato si è posto dal primo momento.

6.12.1 Un singolo protocollo per differenti funzioni

Dai discorsi fatti finora emerge chiaramente che le richieste che il nuovo protocollo dovrebbe soddisfare in termini di comunicazione sono variegate. Da un lato c'è il tipo di comunicazione che dovrebbe avvenire tra due macchine collaboranti allo scopo di sincronizzarsi, dall'altro lato potrebbe emergere la necessità di aggiornare un database inserendo dati di produzione, senza stringenti vincoli di real-time, o addirittura a fine giornata.

Nel primo scenario è necessaria una trasmissione a minima latenza, essendo di fatto un processo real-time, ma potrebbe anche darsi il caso che la perdita di qualche pacchetto sparso non comprometta l'attività in corso. Questo potrebbe essere il caso ad esempio dell'invio di un flusso di dati contenente la velocità di un trattore. Perdere un singolo valore compreso tra due ricevuti quasi sicuramente non sarebbe un problema, ma a contrario una bassa latenza sarebbe indispensabile allo scopo di controllare accuratamente i movimenti relativi di due macchine che lavorano vicine.
Questo genere di pacchetti si possono definire 'isocroni', considerando la notevole importanza dl fattore tempo nel flusso di informazioni.
In un secondo scenario invece durante la trasmissione di dati di produzione possono essere tollerati tranquillamente anche ritardi nell'ordine dei secondi, ma la perdita di un solo pacchetto potrebbe compromettere l'attività di gestione.

6.12.2 Quadro generale

Allo scopo di discutere del soddisfacimento di requisiti così differenti sono stati introdotti nei paragrafi precedenti alcuni argomenti e metodi sia di carattere generale che più specifico. In questo paragrafo si presenta un breve compendio delle caratteristiche più importanti che fanno parte del nuovo protocollo proposto.
L'idea è quella di adottare una tipologia di rete standard ma non deterministica come base, ovvero una rete basata su TCP/IP, per costruire un nuovo modo di comunicare che possa soddisfare alcuni requisiti in termini di massimo ritardo, affidabilità e sicurezza.
In alcuni altri casi invece l'obiettivo può essere anche diverso, ovvero ad esempio di minimizzare l'overhead di trasmissione, allo scopo di minimizzare i costi di trasmissione nel caso di utilizzo di reti 'costose' come quella GSM/GPRS.
Ancora implementazione del nuovo protocollo può avere tra gli obiettivi quello di utilizzare software e hardware standard, in modo da semplificare l'adozione dello Standard e favorirne la diffusione.
Per raggiungere tutti questi obiettivi è importante tenere in conto due concetti fondamentali, ovvero la priorità a livello di rete e di task di sistema all'interno delle unità elettroniche di elaborazione, e anche la organizzazione dei dati all'interno del datagramma del protocollo.
Quest'ultimo deve essere modificato in modo da riflettere differenti requisiti, alcuni dei quali sono stati introdotti. Ad esempio un datagramma utilizzato per sincronizzazione deve

contenere un CRC robusto, e quindi di dimensioni maggiori, ed un piccolo payload, in modo di consentire una veloce trasmissione, mentre per trasferire dati riferibili alla produzione è preferibile un payload di dimensioni molto maggiori in modo da minimizzare l'overhead di protocollo, ma può essere sufficiente un CRC più semplice.
Parlando invece di comunicazioni a bassa latenza sono stati immaginati due differenti scenari a livello di architettura del protocollo.
Ovvero un protocollo molto semplice ottimizzato per una trasmissione a latenza minima, o un protocollo più orientato ai servizi, che possa fornire più funzionalità alle applicazioni.
Nel primo scenario potrebbe implementare semplicemente un canale di comunicazione ad alta priorità e minima latenza, lasciando al livello applicativo tutti i controlli necessari, ad esempio in termini di tempistiche e affidabilità.
Nel secondo scenario invece il protocollo stesso potrebbe fornire servizi predefiniti come un trasporto deterministico dei dati oppure direttamente funzionalità di sincronizzazione tra host.
In ogni caso il protocollo potrebbe essere realizzato mantenendo la compatibilità con la suite TCP/IP, in modo da soddisfare ulteriori necessità di comunicazione grazie ben conosciuti strumenti software standard tipici di questo ambiente. Anzi l'idea di base è di utilizzare quanto più possibile lo stack TCP/IP nelle sue funzioni di base, se possibile incapsulando il nuovo protocollo in modo da mantenere la compatibilità con le applicazioni esistenti.
Per realizzare la trasmissione wireless invece la tecnologia scelta è la famiglia 802.11, a cui comunemente ci si riferisce con il termine Wi-Fi. Ciò per vari motivi in precedenza elencati, ma in particolar modo per il fatto per essa sono disponibili frequenze riservate in tutto il mondo, anche se qualche piccola differenza esiste ad esempio in termini di potenza di trasmissione consentita. Nel seguente paragrafo inizia la discussione del dettaglio del protocollo, partendo da alcune definizioni necessarie che vengono riportate dalla normativa.

6.12.3 Alcune definizioni

Di seguito alcune definizioni mutuate direttamente da standard esistenti, come CAN e ISOBUS per compatibilità, che saranno utilizzate nel seguito.

- Machine: Una unità o serie di unità identificate da un sistema di comunicazione wireless, in grado di stabilire ottenere un indirizzo ed di stabilire una comunicazione in rete.

- Machine Id: nel nostro caso l'indirizzo IP della macchina, ottenuto ad esempio da un server DHCP o definito staticamente in una tabella nella fase di setup della lavorazione.
- Function: Una serie di unità in una macchina destinate ad una specifica funzione così come definito nella documentazione di ISO 11783 (ISOBUS).
- Function Id: Identificatore di funzione così come definito nella documentazione di ISO 11783 (ISOBUS).
- Message: Una serie di dati, comandi, informazioni trasmesse in rete usando un unico messaggio, utilizzando un qualsiasi protocollo come UDP, IP, TCP o altri, in funzione della natura della informazione o del comando da trasmettere.
- Message Id: Un identificatore univoco della informazione o del messaggio, utilizzato nello stesso modo di quanto indicato dalle normative CAN e SAE J 1939 o ISO 11783, per messaggi statici; il protocollo di trasporto in questa fase non viene considerato.

6.12.4 Livelli di priorità

Per soddisfare i requisiti sinora introdotti il protocollo potrebbe implementare otto distinti livelli di priorità. Alcuni livelli di priorità qui definiti non si riflettono direttamente in una priorità di traffico dati, ma potrebbero essere utilizzati per riferirsi a strutture di datagramma differenziate a seconda delle necessità applicative.
Il numero di otto livelli è stato scelto anche per compatibilità con gli otto livelli di priorità esistenti nella norma SAE J 1939 e di conseguenza in ISOBUS (primi 3 bit dei 29 bit del packet Id dello standard CAN 2.0 B). Da notare che alcuni livelli sono stati riservati a futuro utilizzo, in modo da poter inserire alcuni livelli a priorità intermedia nel caso emergessero necessità non previste in fase di prima standardizzazione.
Di seguito l'elenco delle priorità proposte:

1. Level 1 - non utilizzato. Riservato per sviluppi futuri, ad esempio latenza inferiore al minimo previsto dalla prima revisione dello Standard.
2. Level 2 - alta priorità di trasmissione - minima latenza (target =< 100ms). La struttura di datagramma proposta per questo livello si può trovare nel seguito.
3. Level 3 - non utilizzato. Riservato per sviluppi futuri.

4. Level 4 - media priorità di trasmissione - media latenza (target =< 1000ms). La struttura del datagramma per questo livello potrebbe essere simile a quella del livello 2. Questo livello di priorità potrebbe essere usato ad esempio per l'interazione con un operatore.
5. Level 5 - non utilizzato. Riservato per sviluppi futuri.
6. Level 6 - bassa priorità di trasmissione - alta latenza (target =< 5s). Ad esempio sincronizzazione di accessori per gestione flotte, ovvero un accessorio che informa un veicolo circa la sua attuale posizione.
7. Level 7 - non utilizzato. Riservato per sviluppi futuri
8. Level 8 - nessuna priorità di trasmissione - nessun obiettivo in termini di ritardo massimo, latenza dipendente dalla rete. Struttura del datagramma con header minimo, CRC leggero e payload di grandi dimensioni. Incapsulamento in UDP per compatibilità, o anche uso diretto di TCP.

6.12.5 Definizioni su comunicazione ed errori

Per quanto riguarda il sistema di comunicazione vigono le seguenti definizioni (in inglese perché in questa forma ricorrono in normativa):

- Communication System. Un sistema per la trasmissione di messaggi legati o non legati alla sicurezza, e costituito insieme da unità di sistema - sorgenti e destinazioni delle informazioni - da un mezzo o rete di trasmissione (ad esempio linee elettriche galvaniche o in fibra ottica, trasmissione radio) e da tutte le parti elettroniche di interfaccia.
- Message Source (Message Sender). Entità che invia un messaggio.
- Message Sink (Message Receiver). Entità che riceve un messaggio.
- Message. Un messaggio costituito da dati utente, indirizzo ed ulteriori informazioni atte ad assicurare l'integrità delle trasmissione, etc.
- Maximum extension size. Massimo numero consentito di sender e receiver che possono essee coinvolti nello scambio di un singolo messaggio così come definito per il particolare sistema di comunicazione.
- Process safety time. Massimo ritardo accettabile per la ricezione di un messaggio, senza problemi per la sicurezza del sistema.

Per quanto riguarda gli errori di trasmissione vigono le seguenti definizioni (in inglese perché in questa forma ricorrono in normativa):

- Repetition. Un errore dovuto ad un difetto di un componente del sistema di comunicazione , in cui, messaggi vecchi e non aggiornati sono ripetuti ad un istante temporale errato, e che possono causare pericolosi disturbi al receiver (ad esempio segnalare 'porta di accesso chiusa' quando invece è già aperta).
- Loss. Cancellazione non intenzionale di un messaggio dovuta a rottura di un componente del sistema di comunicazione o a malfunzionamento della linea. (pericoloso se ad esempio viene persa una richiesta di immediato stop di sicurezza in una macchina agricola)
- Insertion. Inserzione non intenzionale dovuta a guasto di un componente del sistema di comunicazione.
- Incorrect sequence. Modificazione non intenzionale della sequenza dei messaggi dovuta ad un guasto di un componente del sistema di comunicazione o a problemi di routing. Ad esempio - sequenza corretta: prima dello stop di sicurezza viene ridotta la velocità. Sequenza errata - prima stop immediato di sicurezza, poi riduzione della velocità. Come conseguenza la macchina continua ad avanzare perché ripartita in conseguenza del comando di velocità ridotta.
- Message falsification. Falsificazione di messaggi dovuta ad un errore di un componente del sistema di comunicazione o ad errori del mezzo trasmissivo.
- Retardation. Ritardo non intenzionale che può essere dovuto a:

a) un sovraccarico del percorso di trasmissione nel normale scambio dati

b) al fatto che un componente del sistema di comunicazione causa sovraccarico spedendo messaggi scorretti e ripetuti o a priorità più bassa (problemi di priorità).

6.12.6 Metodi per confinamento di errori di comunicazione

In ISO15998 sono riportati una serie di definizioni e metodi atti a ridurre o evitare errori di trasmissione ed è anche inserita una tabella che consente di ottenere la combinazione di contromisure utili ad ottenere un canale di comunicazione sicuro. La tabella è stata inserita nel successivo paragrafo 6.12.7 - Tabella 6.12 per cui non viene qui ripetuta, ma di seguito è presente la lista di contromisure adottabili.

- Running number. Un numero consecutivo di sequenza è aggiunto ad ogni messaggio scambiato tra sender e receiver. Questo running number può essere definito come un campo dati addizionale che contiene un numero che cambia in maniera predeterminata tra un messaggio e il successivo.
- Time stamp. Il contenuto di un messaggio è di solito valido in un assegnato istante. Il time stamp è una data e ora che sono aggiunte ad un messaggio trasmesso dal sender. Il time stamp si distingue in relative time stamp, absolute time stamp e dual time stamp.
- Relative time stamp. Un relative time stamp è un time stamp che è derivato da un orologio locale di un dispositivo. Normalmente in tal caso non c'è relazione tra i time stamp dei distinti dispositivi.
- Absolute time stamp. Un absolute time stamp è derivato da un orologio comune di riferimento per un gruppo di dispositivi.
- Dual time stamp. Un dual time stamp è creato se due componenti un sistema di comunicazione si scambiano e comparano i propri time stamp. In questo caso i time stamp di ogni singolo componente sono costruiti indipendentemente.
- Time expiration (time-out). Il receiver verifica durante la trasmissione di un messaggio, se il ritardo tra due messaggi consecutivi eccede un valore predeterminato. Quest'ultima eventualità è considerata errore.
- Reception acknowledgement / echo. Il receiver manda indietro al sender un nuovo messaggio con il contenuto del messaggio originale (echo). Per esempio, un reception acknowledgement potrebbe ripetere i dati ricevuti per consentire al sender di verificarne la corretta ricezione. Alcuni sistemi di comunicazione usano termini come 'reception acknowledgement', 'echo' and 'receipt' come sinonimi.
- Identification for message sender and receiver. I messaggi possono contenere una identificazione uniforme del sender e/o del receiver che definisce l'indirizzo logico dei partecipanti.
- Redundancy with cross monitoring. Le informazioni di sicurezza sono trasmesse indipendentemente due volte in messaggi differenti. I messaggi vengono poi monitorati dai receiver in maniera incrociata. Se differiscono vuol dire che è occorso un errore durante la trasmissione oppure in una unità che li ha processati in trasmissione o in ricezione.
- Different data integrity assurance safety-related (SR) and non-safety-related (NSR) data. Se dati relativi alla sicurezza - safety-related (SR) - e non relativi alla sicurezza - non-safety-related (NSR) - sono trasmessi tramite lo stesso canale, differenti principi e verifiche di

integrità dei dati sono utilizzati (differenti algoritmi di CRC, differenti generatori polinomiali) per essere sicuri che messaggi NSR non possono influenzare alcuna funzione legata alla sicurezza in un receiver SR. Nota: differenti verifiche di integrità dati può anche significare che messaggi NSR non sono affatto dotati di verifica di integrità.

6.12.7 Analisi di sicurezza

Data l'importanza che possono rivestire le comunicazioni inter-macchina, soprattutto se destinate alla sincronizzazione di marcia o delle sessioni di lavoro delle macchine, è importante analizzare anche aspetti di sicurezza legati alla struttura della comunicazione ed al livello di confidenza di ricezione di stringhe di dati effettivamente prive di errori.
Come premessa va detto che in ogni caso ogni tipo di comunicazione per ora prevista per la sincronizzazione del moto delle macchine fa riferimento a modelli di sicurezza di tipo 'Fail Silent', come definito dalle normative **[4]** e **[3]**, ovvero a informazioni la cui assenza può condurre la macchina all'arresto totale o all'arresto di alcune funzionalità senza incorrere in problemi di sicurezza; in sintesi la stasi è stato di sicurezza per la macchina. Questo assunto permette di realizzare un link wireless, cosa che sarebbe impossibile in tutti i casi ove non fosse tollerabile la perdita del link di comunicazione.
La possibilità di analizzare il flusso informativo proveniente da sorgenti Wi-Fi remote e di poter determinare se esso è affidabile o meno e, di conseguenza, dichiararne la validità o la invalidità, permette di realizzare sistemi che si affidano alle informazioni remote per l'esecuzione di particolari attività ma che, in assenza di dati remoti, sono in grado di lavorare autonomamente o, in ogni caso, di portare se stessi in condizioni di sicurezza.
In questo lavoro non ci si è occupati delle modalità di sicurezza delle macchine, piuttosto si è incentrata l'attenzione sul fornire strumenti adatti alla corretta valutazione della integrità dei dati acquisiti attraverso il canale wireless.
La necessità di rispondere a rigidi criteri definiti dalle normative di sicurezza generali **[4]** e specifiche **[5]** e **[7]**, permette anche di definire delle strutture dati e delle funzioni di controllo che permettono di affrontare e superare analisi di sicurezza di tipo FMEA (Failure Mode and Effect Analysis) e FTA (Fault Tree Analysis) e di realizzare la copertura completa dei possibili errori di trasmissione e ricezione che potrebbero accadere durante una sessione di scambio dati, anche in presenza del canale di trasmissione.

Quindi, al fine di aumentare la reliability del protocollo e il livello minimo di safety (SIL = Safety Integrity Level) della comunicazione sono implementate le regole descritte in **[6]**. La affidabilità del protocollo è quindi garantita dalle contromisure elencate di seguito, necessarie ad annullare i difetti della trasmissione derivanti dalle più comuni tipologie di errori di trasmissione e ricezione su un canale di comunicazione:

1. Running Number
2. Timeout per tutti i messaggi, accoppiato a un Time Stamp che è trasferito contestualmente al messaggio, al fine di sincronizzare le temporizzazioni di trasmittente e ricevente
3. Un CRC software aggiunto a quelli già embedded nel protocollo nel controller Ethernet
4. Acknowledge esplicito richiesto per ogni messaggio di tipo Safety related

Tali misure salvaguardano contro i più comuni errori in cui incorrono i sistemi di comunicazione e fa riferimento alla Tabella 6.12 di correlazione della copertura degli errori da parte delle contromisure.

Transmission errors/counter measures	**Running Number**	**Time Stamp**	**Time Expectaion**	**Reception Ack**	Sender/ receiver identification	Data integrity assurance	Redundancy with cross check	Different data integrity assurnace system
Repetition	**X**	**X**					X	
Loss	**X**			**X**			X	
Insertion	**X**			**X**	X		X	
Incorrect Sequence	**X**	**X**					X	
Message Falsification				**X**		X	X	
Retardation		**X**	**X**					
Coupling of SR and NSR Information				**X**	X			X

Tabella 6.12: Tabella Errori di comunicazione e contromisure

Va da sè che tale sovrastruttura appesantisce non solo il throughput trasmissivo, ma anche l'impegno di CPU per l'esecuzione di tutte le procedure necessarie alla gestione del protocollo

e deve essere implementata solo in funzione delle criticità delle informazioni contenute nei diversi tipi di messaggi scambiati. Il livello di sicurezza atteso dipenderà quindi dal livello di performance SIL richiesto dal tipo di dato ed è quindi non solo una caratteristica del protocollo ma anche del contenuto informativo in funzione alla tipologia di azione che da esso deriva e dall'ambito applicativo che nel caso in esame è quello di macchine agricole che si muovono in ambiente non strutturato, ma con possibile compresenza umana.
La struttura data al contenuto dei pacchetti è derivata dalla classica struttura dei pacchetti scambiati sulle reti CAN, ed è evidenziata nella seguente Tabella 6.13:

	Priority	Machine Identifier	Function identifier	Sequence Number	Time stamp	Flags	Payload	SW CRC
lunghezza (byte)	1	4	4	2	4	1	576 - altri campi - UDP Header - IP Options	5

Tabella 6.13: Struttura del pacchetto incapsulato in UDP

La scelta è stata fatta al fine di minimizzare il lavoro della CPU per realizzare operazioni di gateway tra le reti di macchina, che sono standard CAN 2.0 B, con protocollo di alto livello SAE J 1939 e ISO 11783 (ISOBUS). Analogamente gli 8 livelli di priorità dei pacchetti sono stati così definiti per essere del tutto compatibili con la definizione delle classi di priorità definite in **[1]** e in **[2]**.

6.12.8 La struttura del pacchetto

In Figura 6.14 è mostrata la struttura del datagramma proposto per il protocollo al livello di priorità 2 descritto nel paragrafo 6.12.4:

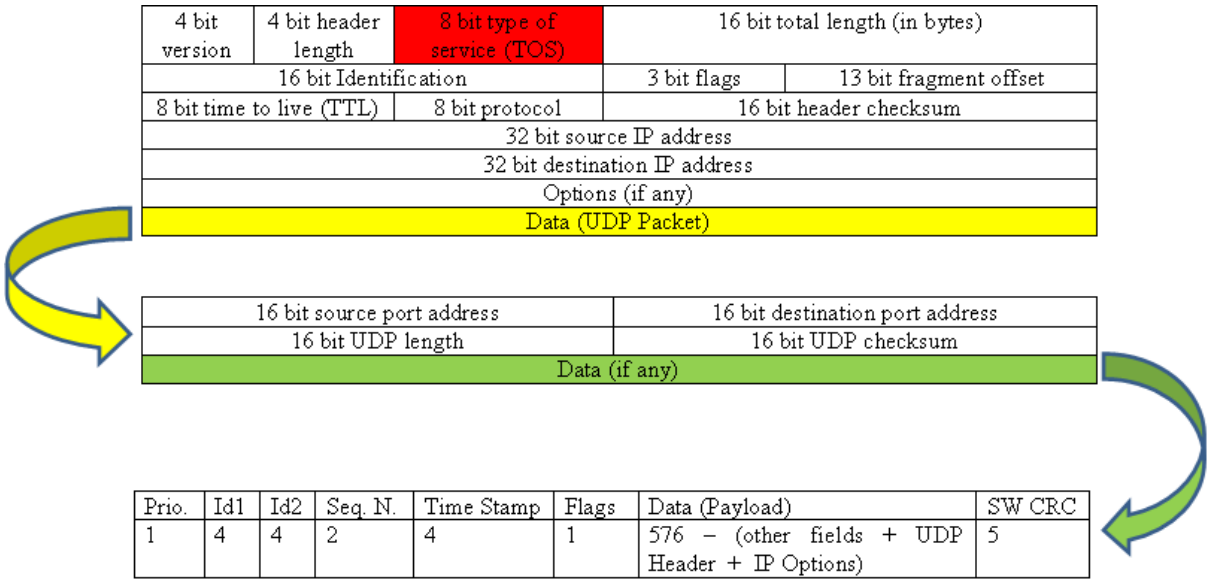


Figura 6.14: Struttura del pacchetto incapsulato in IP e UDP

Partendo dall'alto nella figura è visibile prima il pacchetto formato dal datagramma IP con la sua parte dati evidenziata in giallo. Quest'ultima parte è il datagramma UDP formato da un header e da un payload evidenziato in verde. Infine all'interno del payload UDP è inserito il datagramma del nuovo protocollo proposto.

La struttura del datagramma indicata è stata concepita per trasferire tutte le informazioni necessarie per la sincronizzazione di macchine ed altre funzioni in relazione alla comunicazione a breve raggio ed alta priorità.

Vediamo più in dettaglio i vari campi del datagramma:

- 3 bit (1 byte) - Priority. Per gli otto livelli descritti nel paragrafo 6.2.13 basterebbero tre bit, ma ne sono stati riservati otto per allineamento al byte e sviluppi futuri.
- 32 bit (4 bytes) Machine Id. Come definito dagli standard CAN, SAE J1939 e ISO 11783.
- 32 bit (4 bytes) Function Id. Come definito dallo standard ISO 11783.
- 32 bit (4 bytes)- timestamp con una risoluzione di millisecondi
- 16 bit (2 bytes) - sequence number
- 33 bit (5 bytes) CRC - in standard IEEE CRC32
- 512 byte dimensione massima del payload (576 - 21 - 8 - IP Options) bytes

Per quanto riguarda la dimensione dei pacchetti utilizzando UDP encapsulation, la massima lunghezza di pacchetto totale, quindi calcolando anche la parte IP, deve essere inferiore al 'minimum reassembly buffer size', in tal modo la frammentazione dei pacchetti viene evitata. Per ottenere ciò la dimensione del payload deve essere **[31]** pag. 57:

- 576 byte per IPV4
- 1500 byte per IPV6

Tipicamente molte applicazioni usano 512 bytes per payload UDP onde evitare problemi e overload nel caso fossero presenti le 'IP options' nel datagramma IP (margine di sicurezza).

A causa delle caratteristiche della suite standard TCP/IP per il datagramma del nuovo protocollo utilizzato al massimo livello di priorità per ottenere latenza minima è necessario evitare la frammentazione, e quindi mantenere un payload di piccole dimensioni. Per diversi livelli di priorità dello standard possono invece essere valutate altre dimensioni di payload in funzione del livello di sicurezza e dei tempi di trasferimento massimi richiesti al protocollo . Il tempo di trasferimento in particolare può aumentare notevolmente per payload oltre ad una certa dimensione a cause della frammentazione e conseguente riassemblaggio dei pacchetti effettuato dal protocollo IP.

La struttura del pacchetto è stata generata seguendo le raccomandazioni della norma ISO15998 per la riduzione del tasso di errori residuo nei sistemi di comunicazione.

In conseguenza di ciò tutte le strutture e misure di sicurezza sono state introdotte nel protocollo proposto allo scopo di raggiungere una affidabilità nella comunicazione che sia caratterizzata da un livello di sicurezza pari a quello delle reti CAN, anche se con differenti vincoli temporali e funzionalità disponibili. Ad esempio vista la mancanza di un 'hardware acknowledge' che è presente nei messaggi e viene gestito direttamente in hardware dai controller CAN, un acknowledge esplicito deve essere aggiunto, e deve esser tenuto conto del fatto che l'acknowledge sarà acquisito dal sender dopo un ritardo che non è presente nella comunicazione CAN.

6.12.9 Lo stesso protocollo per scopi differenti

La struttura di base del pacchetto incapsulato in UDP presentata nel paragrafo precedente può essere personalizzata a seconda della funzione che il protocollo deve svolgere. A questo scopo potrebbero essere utilizzati anche dei livelli come definiti nel paragrafo 6.12.4 che non

farebbero riferimento ad una priorità vera e propria ma piuttosto ad una funzione con relativa struttura del datagramma appositamente costruita.
Ad esempio il datagramma di cui al precedente paragrafo è stato concepito per comunicazione real-time utile alla sincronizzazione di macchine durante una lavorazione, e per questo motivo sono stati aggiunti un Running number per verificare la perdita di pacchetti, un Timestamp e una Time Expiration per controllare la corretta ricezione temporale dei pacchetti e i timeout, e un CRC per incrementare la robustezza del protocollo sotto il profilo della integrità dei dati trasmessi.
Questa impostazione rappresenta il livello di sicurezza massimo (almeno fino al punto raggiunto attualmente dalla implementazione del protocollo) richiesto dalle applicazioni: comunicazione Isocrona con requisiti stringenti di sicurezza. Altri tipi di comunicazione possono essere derivati da tale modello, ma anche rendere convenienti scelte differenti.
In caso di comunicazioni per gestione flotte ad esempio le necessità diverse porterebbero a piccoli pacchetti senza eccessive richieste in termini di sicurezza, meno vincoli temporali ma con notevole importanza della integrità dei dati. Sarebbe inoltre probabilmente rilevante la questione dei costi di comunicazione, a causa del bisogno continuo di trasferire dati (ad esempio un pacchetto ogni 30 o 60 secondi per ogni veicolo della flotta) durante le lavorazioni, e quindi anche facendo uso di collegamenti costosi, come GSM/GPRS; un pacchetto che soddisfi esigenze di questo tipo potrebbe essere realizzato sfruttando direttamente UDP, che offre un CRC leggero con un minimo overhead di protocollo e che consente di realizzare una comunicazione da uno a molti tramite un unico gateway collegato ad Internet grazie ad una unica linea che consente il dialogo tra vari veicoli e server geograficamente distanti.
Aggiungendo semplicemente un running number il costo della comunicazione sarebbe minimizzato, l'integrità dei dati sarebbe assicurata ad un certo livello dal CRC di UDP, ed il flusso di dati potrebbe essere controllato utilizzando il running number. Il vantaggio di utilizzare un protocollo ancora incapsulato in UDP è la compatibilità con tutte le implementazioni di TCP/IP. In pratica potrebbero essere utilizzate direttamente le funzionalità offerte da TCP/IP.
Ancora un altro tipo di comunicazione è quella richiesta dal trasferimento di file raccolti nelle memorie di massa di unità embedded a bordo di veicoli, ad esempio per data logging. In tal caso grandi quantità di dati dovrebbero essere trasferite dalle unità dedicate al data logging verso il server della azienda agricola ad esempio, o di proprietà del costruttore del veicolo, in

funzione della natura dei dati stessi. Dati di questa natura potrebbero essere trasferiti tramite una connessione Wi-Fi una volta che il veicolo è rientrato alla base operativa, con costi di trasferimento minimi e senza appesantire la rete nel momento in cui deve essere gestita anche la sincronizzazione real-time, ovvero durante la lavorazione. Oppure in caso di necessità , potrebbero essere trasferiti in fase di lavorazione al server o ai server usando una connessione dedicata GSM/GPRS/UMTS/3G tramite un modem collegato ad Internet senza rischi per la coesistenza del traffico di sincronizzazione, però aumentando i costi. In questi ultimi casi sarebbe possibile utilizzare anche un qualsiasi protocollo preesistente basato su TCP/IP (ad esempio FTP or SFTP).
Dunque a causa dei vari metodi utilizzabili per la gestione della priorità e della banda disponibile, l'idea è quella di creare in realtà non un singolo protocollo, ma un set di protocolli unificato grazie ai quali una comunicazione short range a bassa latenza possa convivere con altri tipi di comunicazione per data logging, fleet management, programmazione di lavorazioni, comunicazioni audio e video, traffico generico a più bassa priorità.

6.12.10 Un set di protocolli - differenti funzionalità

In realtà esistono anche modi diversi dal punto di vista delle funzionalità offerte di realizzare un nuovo set di protocolli che riescano a soddisfare le attuali richieste emergenti dal settore delle macchine agricole, che si possono comprendere tra due estremi opposti:

1. Il primo è di realizzare un set di protocolli che includa solo funzioni di comunicazione di base; in tal caso tutte le funzionalità ed i servizi relativi alla gestione delle priorità, al controllo e verifica di timestamp e timeout, alla gestione dei CRC, allo scheduling di messaggi, alla regolazione del traffico e della banda, sono lasciati alle applicazioni, e quindi ogni costruttore può implementare tutte queste funzioni come desiderato. Il protocollo risultante sarebbe molto 'leggero' e necessiterebbe di una minima documentazione.
2. Il secondo modo prevede un set di protocolli che regola e standardizza tutte le funzioni di comunicazione e quelle responsabili di tutti i servizi al punto precedente elencati; il set di protocolli risultante sarebbe molto più 'pesante', ma molto probabilmente garantirebbe una maggiore compatibilità tra le macchine ed un traffico meglio regolato, più sicuro, ma con alcuni vincoli di throughput, a causa dei controlli implementati. Sarebbe inoltre prob-

abilmente necessaria una fase di sperimentazione applicativa estesa per tarare tutti i meccanismi citati nel mondo reale.

La prima soluzione lascia ai progettisti la libertà di definire i timing delle varie attività e ad esempio di allocare la banda disponibile per i differenti protocolli, con il vantaggio di adattare i timing di comunicazione alle necessità della specifica applicazione, ma porta con sè anche una incertezza di prestazioni quando si consideri il lavoro contemporaneo di altre macchine e dispositivi progettati da altri.

La seconda possibilità al contrario regola tutto il traffico quindi garantisce compatibilità e prestazioni per tutte le macchine che partecipano alle attività.

Tra i due estremi evidenziati sono ovviamente possibili tutte le soluzioni intermedie.

Per terminare il lavoro iniziato durante il corso del presente Dottorato perciò è necessario che vengano svolte le attività necessarie a terminare la definizione degli obiettivi del nuovo set di protocolli ed inoltre che ci sia un contributo alla implementazione ed ai test indispensabili per definire completamente i confini di applicazione e i limiti delle prestazioni e funzionalità del nuovo set di protocolli da parte di ISO.

6.13 Ambiente di sviluppo

L'ambiente di sviluppo e test utilizzato nel caso del Task 2 è in realtà molto simile a quello utilizzato nel corso del lavoro per il Task 0, per cui onde evitare ripetizioni si rimanda al Capitolo 4 per una descrizione dei suoi componenti fondamentali, ed in particolare ai paragrafi 4.7 , 4.7.1 , 4.8 , 4.8.1 , 4.8.2 , 4.10.1 , 4.11.2.

Nei paragrafi successivi saranno introdotti alcuni strumenti e componenti non utilizzati nel corso del Task 0 e che dunque rappresentano degli elementi di novità, che però sono assolutamente coerenti con lo spirito di fondo che ha animato tutte le scelte compiute durante il lavoro di Dottorato.

6.13.1 Subversion

Subversion è un sistema di controllo versione Open Source. Fondato nel 2000 da CollabNet, Inc., il progetto Subversion ed il relativo software hanno avuto un incredibile successo negli ultimi anni, tanto da essere diffusissimamente adottato in molti ambienti, dal mondo dell'Open Source, a quello accademico fino ad affermarsi anche in grandissime società. Di recente il progetto è entrato a far parte della Apache Software Foundation **[70]**.

In informatica, il controllo versione è la gestione di versioni multiple di un insieme di informazioni. Viene usato prevalentemente nello sviluppo di progetti informatici per gestire la continua evoluzione dei documenti digitali come il codice sorgente del software o la documentazione testuale su cui può lavorare una squadra di persone. Le modifiche a questi documenti sono identificate incrementando un numero o un codice associato ad essi, denominato "numero di versione", "etichetta di versione", o semplicemente "versione", e sono anche associate all'utente del sistema ha apportato la modifica.
Nel corso del presente Dottorato Subversion server è stato utilizzato nello sviluppo di tutti i software di una certa complessità adoperando come client un plugin di Eclipse, conosciuto ambiente integrato di sviluppo (IDE) introdotto al paragrafo 4.8.2.

6.13.2 Octave

Altro strumento introdotto nel corso del Task 2 è Octave. GNU Octave è un linguaggio interpretato ad alto livello, il cui utilizzo principale è il calcolo numerico, che viene distribuito con una licenza GPL, ed è quindi software libero **[71]**.
Octave fornisce strumenti utili alla soluzione per via numerica di molti problemi matematici, oltre che per il trattamento di informazioni in generale. Oltre a ciò fornisce potenti funzionalità grafiche per la visualizzazione dei dati. Octave può essere usato tramite la sua interfaccia a linea di comando interattiva, ma anche scrivendo veri e propri programmi non interattivi.
Oltre a ciò il linguaggio di Octave è simile a Matlab di The MathWorks Inc. per cui molti programmi sono facilmente portabili tra le due piattaforme.
Con Octave sono stati scritti tutti i software di post-elaborazione dei test e generazione dei grafici, che saranno descritti più in dettaglio nel seguito.

6.14 Strumenti software sviluppati: quadro generale

Durante lo svolgimento del Task 2 è stato necessario sviluppare una serie di software che implementassero il protocollo vero e proprio in tutte le successive revisioni e varianti, che alcune applicazioni che ne simulassero l'uso. Oltre a ciò è stata progettata e realizzata una intera suite di strumenti software che consentono di effettuare test in maniera completamente automatizzata, e acquisiscono di una serie di dati in tempo reale salvandoli su alcuni file. In un secondo momento altri strumenti della suite consentono di effettuare un post-processing dei dati e di generare una serie di grafici e di report sui test effettuati, in maniera automatica ma

con molti parametri configurabili, in modo da evidenziare alcuni istanti temporali, oppure variare la risoluzione dei grafici, generare output dettagliati per il debug e la verifica degli algoritmi.

In Figura 6.15 è mostrato uno schema della architettura generale della suite di software sviluppata:

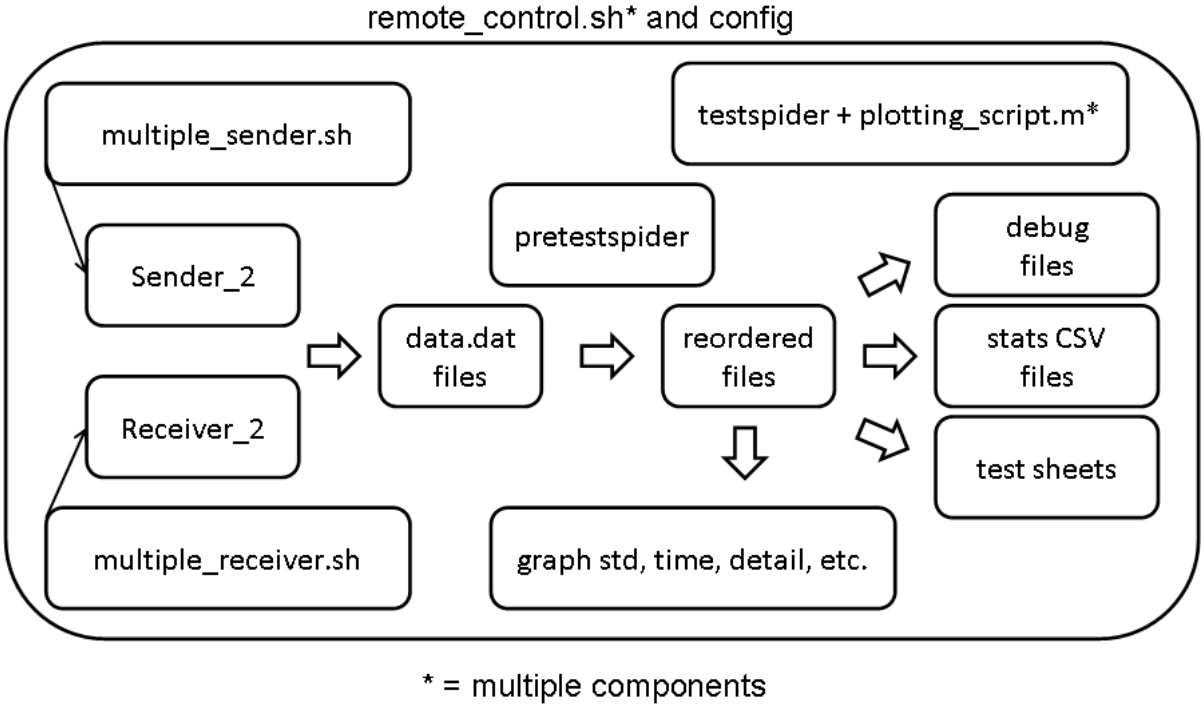


Figura 6.15: Schema generale della suite software sviluppata

Per dare una idea complessiva del lavoro svolto i software sono stati sviluppati usando come linguaggi C standard per la programmazione di sistema, scripting Bash per script di servizio, automatizzazione dei test e post-processing dati, e il linguaggio Octave per le applicazioni di post-processing dei dati, calcoli statistici e generazione dei grafici. Le righe di codice sviluppate ad oggi sono svariate migliaia.

I file di output alla fine del processo di test ed elaborazione dei risultati contengono tutte le elaborazioni intermedie per consentire verifica dei calcoli e del processamento dati nel caso venga attivata l'opzione di debug. Vengono inoltre generati file in formato CSV contenenti tutti i dati statistici calcolati per ogni test. Ancora vengono generati dei test sheet contenenti le configurazioni hardware e software di tutte le macchine coinvolte e e tutti i parametri passati ai vari software in modo da salvare in maniera completamente automatica e trasparente all'utente tutte le informazioni che consentono di risalire all'ambiente di test utilizzato per una

prova in particolare. Oltre a ciò vengono generati grafici con un gran numero di parametri personalizzabili, o auto-configuranti.
Nei successivi paragrafi sono presenti brevi cenni al funzionamento dei software della suite.

6.14.1 Software in C

Il linguaggio C è stato scelto per implementare tutte applicazioni a livello di sistema, per la sua velocità ed efficienza. Le sue prerogative si sono rivelate essenziali nel caso dei software che implementano il protocollo e simulano lo strato applicativo, visto che si intende misurare prestazioni real-time con tempi dell'ordine di grandezza dei millisecondi.

6.14.1.1 Sender_2

Software in C per la generazione di pacchetti e simulazione di protocollo isocrono realtime. Data la sua complessità è stato sviluppato utilizzando l'IDE Eclipse.
Di seguito sono mostrate alcune delle opzioni accettate in ingresso per dare una idea delle sue funzionalità.
Usage:
Sender_2 [-n[pkt_number]] [-r[-20:+19]] [-Pport_number] [-ddelay] [-tp] source_ip target_ip
- n[value] number of packets to send between 1 and 100000. Default value is 100.
- r[-20 to +19] set the process to be reniced by value. Default value is -20
- p set the TOS priority for the socket.
- l enable syslog logging, it will send to the dhcp server useful logs.
- t enable some useful features in realtime environments.
- d[0 to 90] set the delay on exit in seconds. Default value is 2s.
- w[0 to 100000] set the periodical sending interval (in us). Default value is 10ms.
- P[0 to 65535] specify port to bind. Default is 5000.
- N["string"] notes (max 255 char)

6.14.1.2 Receiver_2

Software in C multithreaded per la ricezione di pacchetti e simulazione di protocollo isocrono realtime. Data la sua complessità è stato sviluppato utilizzando l'IDE Eclipse.
Di seguito sono mostrate alcune delle opzioni accettate in ingresso per dare una idea delle sue funzionalità.

Usage:
Receiver_2 [-rltp] dest_ip
- r[-20 to +19]set the process to be reniced by value. Default value is -20.
- l enable syslog logging. Will send to the syslog server useful logs.
- t enable some useful features in realtime environments.
- p[0 to 65535] specify port to bind. Default is 5000.
- N["string"] notes (max 255 char).

6.14.2 Octave scripts

Gli script Octave occupano una posizione importante all'interno della suite, poiché implementano tutta la fase di post-processing dei dati, calcolo statistico e generazione grafici. Infatti non avrebbe senso far effettuare ai sistemi sotto test tutte le elaborazioni necessarie, visto che sicuramente non verrebbero effettuate a bordo di una macchina agricola.

6.14.2.1 plotting_script.m

Script Octave che naviga la cartella corrente, ordina e carica i dati. In seguito ad alcune verifiche sui dati in ingresso (ad esempio reordering) calcola tutti i dati statistici ed elabora su richiesta vari tipi di grafici in maniera completamente automatica. E' costituito da alcune migliaia di righe di codice ed è dotato di svariate funzionalità, tra le quali:

- modalità debug: scrive a video e su disco tutti i risultati intermedi dei calcoli effettuati in modo da consentire verifica di funzionamento
- test sheet: genera automaticamente file contenente informazioni dettagliate sulla configurazione di test
- standard view: grafici standard (flusso intero) con differenze tra tempi di arrivo dei pacchetti
- detail view: grafici di dettaglio con finestratura regolabile
- time view: grafici con riferimenti temporali - sia standard che detail view

Vi sono molti altri parametri personalizzabili, alcuni esempi di output sono nella sezione dedicata ai test.

6.14.2.2 generate_test_sheet.m

Invocato da plotting_script è un parser degli header nei file di dati che ricava automaticamente parametri circa il test attuale raccolti dalla catena di software precedenti e genera automaticamente un file di output (test_sheet.txt) che descrive in dettaglio la configurazione (configurazione hardware, configurazione di sistema, indirizzi IP, host coinvolti, tipo di test, numero e direzione dei flussi etc.)

6.14.2.3 header_def.txt

File di definizione e configurazione per la gestione dinamica di label nel test_sheet.txt.

6.14.3 Script di Shell

La shell offre tutti gli strumenti per creare script che interagiscono con il sistema, sfruttandone le potenzialità in maniera molto semplice e spesso sorprendentemente efficace. Grazie alle funzionalità della potente shell Bash sono stati realizzati vari script utili, ed anche il potente software che automatizza completamente l'esecuzione di test su un numero arbitrario di macchine. Di seguito brevi cenni sui vari script realizzati.

6.14.3.1 multiple_sender.sh

Wrapper script. Lancia più occorrenze di Sender_2 replicando i parametri in ingresso, ma variando alcuni parametri in maniera automatica, ad esempio il numero di porta per ottenere trasmissioni multi flusso

6.14.3.2 multiple_receiver.sh

Wrapper script. Lancia più occorrenze di Sender_2 replicando i parametri in ingresso, ma variando alcuni parametri in maniera automatica.

6.14.3.4 pretestspider

Spider per il pretrattamento dei file contenenti risultati del test. Naviga ricorsivamente le cartelle con riferimenti temporali validi e raccoglie tutti i file di dati in una unica directory per facilitare ulteriori analisi

6.14.3.5 testspider

Spider per il trattamento dei file contenenti risultati del test. Naviga ricorsivamente le cartelle e lancia opportunamente gli script Octave che operano l'analisi vera e propria dei risultati.

6.14.3.6 remote_control.sh

Script che automatizza completamente una o più sessioni di test su un numero arbitrario di macchine remote, caricando topologia, caratteristiche dei test e configurazioni da appositi file di configurazione (ad esempio hosts.in definisce la topologia della rete). Consente di lanciare sessioni batch di test al variare di ogni parametro, salvando tutti i risultati e le analisi per successiva consultazione, oltre a messaggi in uscita ed errori dei software lanciati. E' stato necessario svilupparlo per realizzare test che coinvolgono un numero di macchine/flussi elevato.

6.15 Un ambiente di sviluppo e test distribuito

L'ambiente di sviluppo e test progettato e realizzato durante il lavoro del Task 2 è costituito in realtà da un insieme di siti fisici interconnessi tramite Internet.

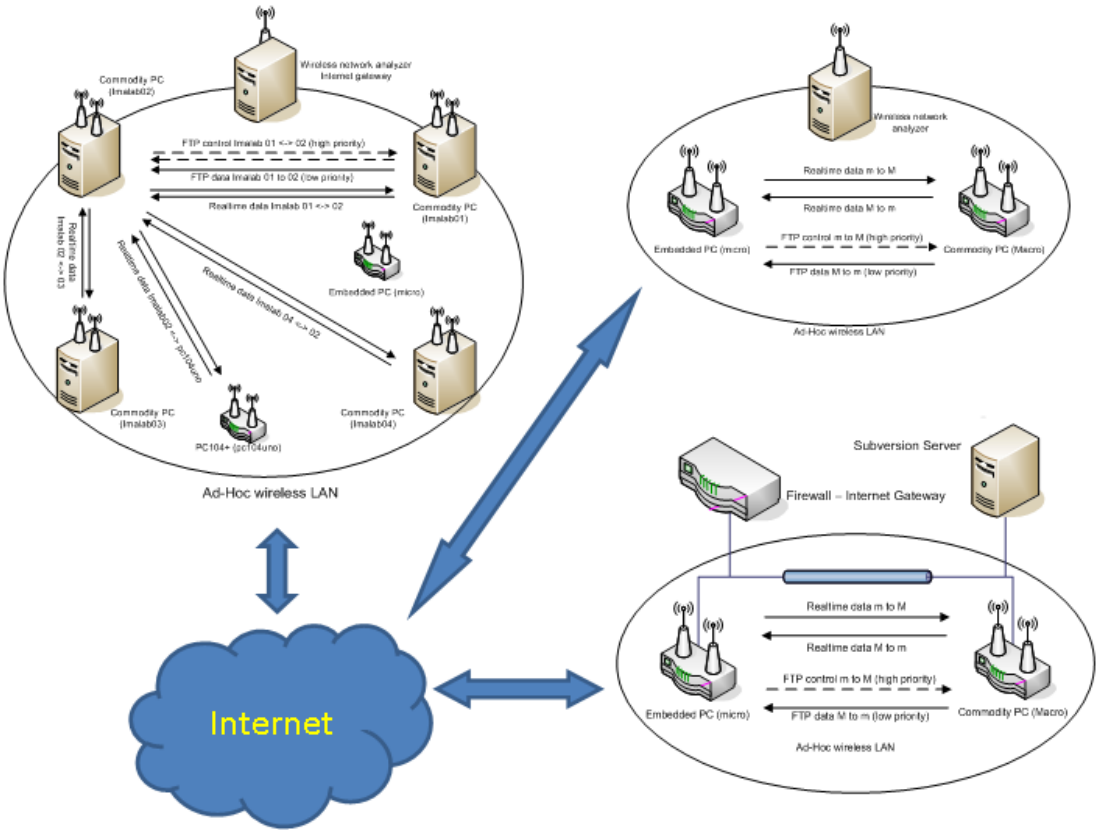


Figura 6.16: Ambiente di sviluppo e test distribuito

Alcuni di questi siti sono permanenti, altri, ad esempio quelli utilizzati per prove outdoor, sono stati realizzati e poi smantellati.
In Figura 6.16 è presente uno schema che dà una idea della struttura complessiva realizzata:In figura è mostrato in alto a sinistra il laboratorio Imalab allestito presso la sede di CNR - Imamoter Ferrara , che è in connessione con un sito di test indoor remoto in alto a destra e con altro sito dove risiedono infrastrutture di rete e server di versioning.
Nei paragrafi successivi sono brevemente descritti alcuni dei siti che compongono l'ambiente, per dare una idea delle condizioni di test del nuovo protocollo.

6.15.1 Imalab

Imalab è un laboratorio di sviluppo e test progettato e realizzato nel corso del presente Dottorato presso Imamoter - CNR. L'Imamoter è un Istituto di ricerca del Consiglio Nazionale delle Ricerche (C.N.R.), ha sede a Ferrara e conta una Unità Organizzativa di Supporto (U.O.S.) presso l'Area di Ricerca del C.N.R. di Torino **[72]**.
Il laboratorio Imalab è rappresentato schematicamente in Figura 6.17:

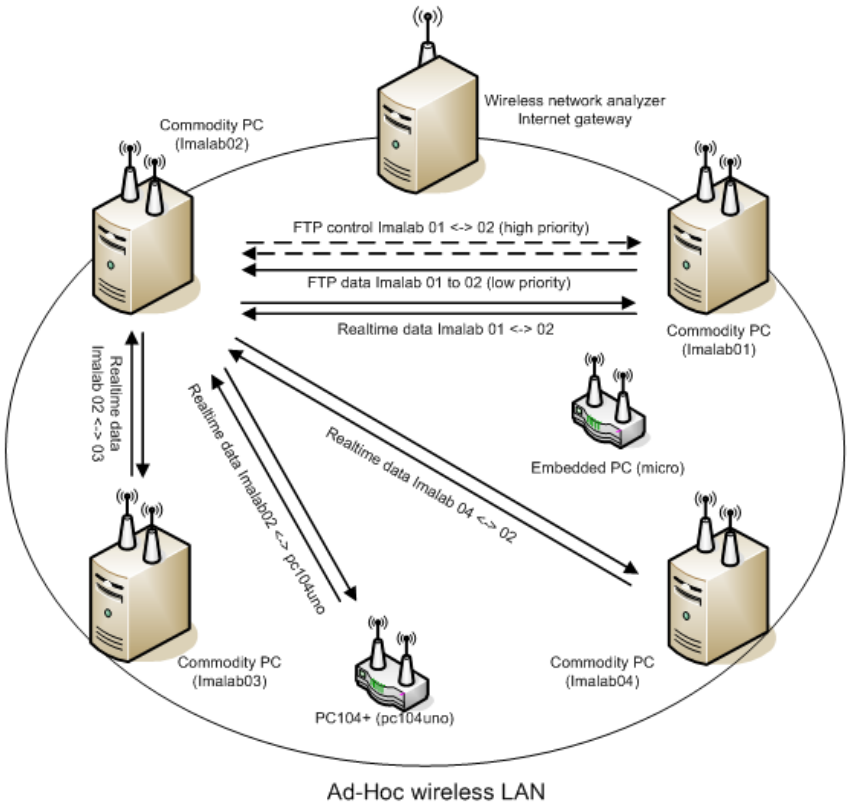


Figura 6.17: Schema Imalab

Per dare idea delle infrastrutture realizzate ed utilizzate per i test di seguito sono forniti alcuni cenni sulle caratteristiche degli elementi rappresentati in figura:

Imalab01-04 commodity hardware PC
Intel pentium 4 3Ghz CPU
CentOS 5.x Linux
- Kernel distro standard
- Kernel da kernel.org con RT patch e/o IMQ patch
Atheros PCI or Mini-PCI 80211n WiFi con adattatore Mini-PCI - PCI.
- Diversity/Mimo con doppia antenna +5Db

Embedded PC (micro)
Embedded PC prodotto da Lex Computech
VIA C7 nano 1Ghz CPU
CentOS 5.x Linux
- Kernel distro standard
- Kernel da kernel.org con RT patch e/o IMQ patch
Atheros PCI or Mini-PCI 802.11n WiFi
- Diversity/Mimo con doppia antenna +5Db

PC104+ (pc104uno)
Embedded PC con scheda SECO104-CX700M
VIA EDEN Ultra Low Voltage 1 GHz CPU
CentOS 5.x Linux
- Kernel distro standard
- Kernel da kernel.org con RT patch e/o IMQ patch
Intel BG 2945 WiFi
- Diversity con doppia antenna +5Db

Network analyzer commodity hardware PC
Intel pentium 4 3Ghz CPU
CentOS 5.x Linux
- Kernel distro standard

Atheros PCI 802.11n WiFi

Nella Figura 6.18 è presente una immagine composta da alcune foto recenti di Imalab:

Figura 6.18: Foto Imalab

6.15.2 Siti di test interni

Allo scopo di effettuare prove in condizioni ambientali diverse sono stati allestiti altri siti di test in locazioni diverse, alcuni permanenti ed altri realizzati ad-hoc. Alcuni siti sono stati utilizzati per prove a distanza variabile di 1, 5, 10 ,50 e 100 metri, essendo dotati di postazioni mobili.

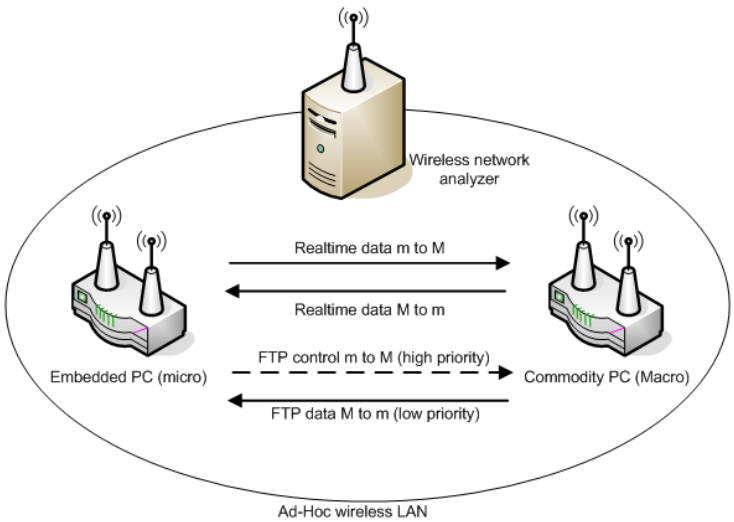


Figura 6.19: Schema indoor test site

In altri casi si sono effettuati test in ambienti elettromagneticamente rumorosi. Lo schema di questo tipo di installazioni è in Figura 6.19.
In tal caso il numero di nodi è minore che nel caso di Imalab, tipicamente sono stati utilizzati due nodi e un analizzatore di rete. Le caratteristiche delle macchine utilizzate sono simili al caso di Imalab, e quindi non vengono qui ripetute, a parte Macro, una macchina particolare con CPU a basse prestazioni utilizzata in vari siti di test.

Macro (commodity hardware PC)
Intel pentium 3 933Mhz CPU
CentOS 5.x Linux
- Kernel distro standard
- Kernel da kernel.org con RT patch e/o IMQ patch
Intel BG 2200 WiFi
- Diversity con doppia antenna +5Db

Nella Figura 6.20 è presente una immagine composta da alcune foto di un sito di test interno permanente realizzato presso una struttura esterna:

Figura 6.20: Foto indoor test site

6.15.3 Siti di test esterni

Allo scopo di effettuare prove in movimento sono stati allestiti siti di test in locazioni all'aperto. Sono stati quindi effettuati test a distanza variabile tra 0 e 200 metri, ed a velocità variabile tra 0 e 40 km/h, condizioni che ben rappresentano la situazione operativa di veicoli che eseguono lavorazioni agricole.

Lo schema di questo tipo di siti di test è concettualmente molto simile a quello mostrato in Figura 6.19 per siti di test indoor e non viene quindi ripetuto. Anche le caratteristiche ed il numero delle macchine utilizzate sono simili al caso di sito di test indoor.

Nella Figura 6.21 è presente una immagine composta da alcune foto di un sito di test esterno:

Figura 6.21: Foto outdoor test site

6.15.4 Piattaforme di test

Nell'arco del lavoro svolto per il presente Dottorato sono stati effettuati un rilevante numero di test utilizzando diverse piattaforme hardware e software, che non sono state evidenziate nella

descrizione dell'ambiente o dei siti di test. In questo paragrafo è presente un elenco di strumenti, piattaforme e tecnologie che può dare una idea del lavoro svolto.

Distribuzioni Linux

CentOS varie versioni – Ubuntu varie versioni – Scientific Linux 5.x - EasyPeasy 2.0 Linux

- Kernel distro default varie versioni
- Kernel con patch RealTime, IMQ ed entrambe varie versioni
- Kernel con patch Agro Kernel

Piattaforme x86 based e non (CPU)

CPU Intel da P3 700Mhz a P4 3Ghz, Core 2 vari, Atom vari

Via Technologies VIA Eden, VIA C7

Embedded CPU (es. Icop Vortex86)

MicroChip dsPIC

Wireless Network (802.11gn – diversity - Mimo)

Intel Wireless 2200BG, 2945BG, WiFi Link 5300 (Mini-pci)

Atheros 5414 (Mini-PCI), 922x (Mini-PCI,PCI), 9222 (Usb)

Intersil Prism (Mini-PCI)

RALink RT2561/RT61 (PCI)

6.16 Ambiente di test

La pianificazione dei test è stata effettuata considerando tutti i differenti tipi di traffico e le condizioni che potrebbero essere rappresentativi di un contesto reale, ad esempio del traffico real-time per la sincronizzazione di macchine e contemporaneamente traffico a più bassa priorità con notevole occupazione di banda, come una trasmissione file su protocollo FTP, o una comunicazione TCP per fleet management o data logging attraverso Internet. Le prestazioni del protocollo nelle sue varie versioni sono state analizzate in molti ambienti differenti. Come evidenziato nella parte sui siti di test sono state effettuate prove in laboratorio, in siti al coperto con varie condizioni ambientali e numero di macchine, e in siti all'aperto variando distanza e velocità dei veicoli coinvolti. Il tipo di test effettuati è stato vario, in Appendice C Figura C.3 è stato inserito un foglio di lavoro utilizzato prima della

completa automatizzazione della procedura per registrare alcuni dati di prove effettuate sul campo per dare una idea delle tipologie di test effettuati. Per generare traffico con elevata occupazione di banda sono stati utilizzati client FTP standard sia a linea di comando che grafici e sotto vari sistemi operativi, Open Source o proprietari. Come server FTP è stato utilizzato VSFTPD – Very Secure FTP Daemon. Il traffico FTP è stato generato trasferendo file da un nodo all'altro, in modo che la velocità di trasferimento fosse limitata solo dalla capacità del link wireless. Le priorità di FTP client e server non sono mai state modificate, ma lasciate pari a quelle di default del sistema operativo.

Una tipologia tipica di test ad esempio prevedeva la trasmissione e la ricezione da parte delle applicazioni sviluppate ad-hoc di 10.000 pacchetti UDP con un periodo di 10ms al variare di condizioni al contorno. Alcune tra le condiziono variate sono priorità più alta sulla applicazione sender o sulla receiver o su entrambe, abilitazione o meno di Type of Service (TOS) IP low delay sul traffico UDP 'real-time', presenza o meno di traffico FTP simultaneo, presenza di rumore elettromagnetico, variazione della distanza o della velocità, variazione del numero di macchine coinvolte, variazione del numero di 'flussi' per ogni macchina, direzione dei flussi stessi.

I sistemi operativi dei vari host sono stati spesso intenzionalmente non ottimizzati ma lasciati nelle stesse condizioni di default a seguito della installazione, per realizzare delle condizioni di caso peggiore. In qualche caso invece si è ottimizzata la configurazione del sistema per verificare le prestazioni in condizioni più vicine alle ottimali. In tal caso ad esempio sono stati disattivati i servizi di sistema non essenziali, ed anche selinux e il firewalling, ed i sistemi sono stati portai ad init 3 in modo da disabilitare l'interfaccia grafica.

Sono state effettuate inoltre prove con varie versioni del kernel mainstream e con varie distribuzioni, ed inoltre utilizzando kernel a cui è stata applicata patch real-time e/o IMQ, in modo da verificarne l'effetto sui test. Nel caso di utilizzo di patch real-time il kernel è stato compilato disabilitando l'Advanced Power Management (APM) e la Advanced Configuration and Power Interface (ACPI) così come consigliato dalla documentazione per evitare aumenti del tempo di risposta. L'hardware utilizzato è stato scelto tra quello compatibile in base al supporto fornito dal kernel Linux. In particolare per quanto riguarda l'hardware di rete sono state testate un certo numero di schede dotate di differenti chipset, comprese alcune compatibili con lo standard 802.11n. Come modalità di connessione Wi-Fi è stato scelto il modo ad-hoc, visto che non prevede la necessità di una stazione base come invece accade alla modalità infrastruttura. Tutti i test sinora effettuati sono stati fatti utilizzando al massimo

connessioni in standard 802.11g, in quanto durante il setup dei sistemi è stato riscontrato che il supporto alla tecnologia ‘n’ nei drivers atheros e in adhoc mode è ancora immaturo. La qualità del link wireless per le prove in laboratorio a breve distanza è stata sempre almeno eguale al 90% come segnalato dai driver delle schede di rete sotto Linux. Nei test a distanza crescente si è cercato di mantenere uniforme la qualità riportata del link a parità di distanza. Quindi tutte le unità durante i test sono state connesse in una rete wireless in adhoc mode per trasmettere i dati oggetto dei test e tramite una connessione ethernet ove necessario per controllare il processo di prova ed acquisire i risultati.

Per quanto riguarda le applicazioni che effettuano test in tempo reale esse sono divise in due parti, un processo sender ed un processo receiver. Il processo sender è stato configurato durante i vari test per girare a diversi livelli di nice da quello default di sistema a -5 e a livelli di priorità real-time dal default a -89, che è il massimo livello di priorità suggerito per applicazioni utente. Infatti le priorità da 90 a 99 sono normalmente utilizzate per processi di sistema, e con i quali nella documentazione della patch RT è raccomandato di non interferire. Nel caso di utilizzo di kernel real-time sono state utilizzate specifiche funzionalità, oltre alla priorità RT ad esempio si è sfruttata la possibilità di ottenere un lock dei processi real-time in memoria in maniera da evitare swap su disco che potrebbe introdurre notevoli ritardi. Una configurazione simile è stata realizzata anche per il software receiver. Quest’ultimo è una applicazione multithreaded composta da due thread che possono girare a differenti livelli di priorità per assicurare la gestione della comunicazione in real-time al livello di ricezione dei messaggi.. Il thread a più alta priorità è stato fatto girare a diversi livelli di nice da quello default di sistema a -5 e a livelli di priorità real-time dal default a -89, esattamente come il processo sender; questo processo nella fase iniziale e finale della sessione di comunicazione è responsabile per il riconoscimento di pacchetti di start e di stop e per la inizializzazione della sessione di comunicazione stessa, ed inoltre attiva un secondo thread a priorità tipicamente più bassa. Durante la fase di comunicazione a runtime gestisce la ricezione dei pacchetti dalla socket di rete e salva tutti i dati utili in un buffer software in memoria. Il thread a priorità più bassa, tipicamente configurato al livello di priorità di default di sistema per le applicazioni utente, una volta inizializzato crea alcuni file su cui scrive varie informazioni sulla configurazione di sistema e di rete, e salva i dati utili dal buffer in memoria sul disco rigido o memoria di massa; alla fine della sessione di comunicazione si occupa poi di chiudere tutti i file aperti, e chiude ordinatamente la sessione inserendo anche altri dati statistici, di controllo e sulla configurazione del sistema.

In seguito vengono poi utilizzati tutti gli strumenti di post-processing per verificare i dati salvati, calcolare dati statistici e generare test sheet e grafici.

Per quanto riguarda i dati rilevanti ai fini della valutazione delle prestazioni del nuovo protocollo implementato e del sistema di comunicazione in generale, ne sono stati valutati vari tipi. Uno di fondamentale importanza è l'intervallo di tempo che intercorre tra due pacchetti di dati successivi sia al momento della generazione che della ricezione. Per verificare questo dato timestamp ad alta precisione sono salvati sia nel sender che nel receiver per ogni pacchetto dati. Nel primo caso il timestamp viene rilevato all'istante in cui il pacchetto stesso viene generato nella macchina sender, allo scopo di includere l'effetto del sistema operativo nella valutazione delle prestazioni, ed è poi incluso nel payload del pacchetto spedito, per essere poi letto e memorizzato su file all'atto della ricezione del pacchetto stesso, insieme al timestamp generato nell'istante di ricezione dal receiver.

La sincronizzazione del tempo tra le varie macchine è basata sul protocollo Internet NTP, utilizzando lo stesso timeserver per tutte le macchine; in ogni caso anche se non si può essere certi che gli orologi di tutte le macchine siano perfettamente sincronizzati, i dati rilevati ai fini del test sono differenze relative tra istanti di ricezione calcolate utilizzando lo stesso orologio, sia esso del sender o del receiver, per cui sono differenze tra dati omogenei e non affetti da differenza di clock o errori assoluti di orario.

I parametri e dati rilevanti per definire le prestazioni del sistema che sono stati considerati nei test sono ad esempio:

- Numero di macchine nel gruppo di test, che definiscono la probabilità di collisione a causa di difetti di arbitraggio nel canale di comunicazione
- Numero di flussi di dati tra una macchina e l'altra nel gruppo di test, che definiscono il livello di traffico e ancora la probabilità di collisioni.
- Numero di pacchetti persi che definiscono la qualità della trasmissione, e gli effetti del traffico sulle macchine in ricezione.
- Differenze temporali tra due pacchetti consecutivi inviati dal sender definite come $\Delta Tsj(i) = Tsj(i) - Tsj(i-1)$ dove S è il Sender, j è l'indice del flusso ed i è il running number del pacchetto.
- Differenze temporali tra due pacchetti consecutivi ricevuti dal receiver definite come $\Delta Trj(i) = Trj(i) - Trj(i-1)$ dove R è il Receiver, j è l'indice del flusso ed i è il running number del pacchetto.

Le differenze così come descritte sono state utilizzate per calcolare tramite Octave, elaborando direttamente i file di output delle applicazioni generati a runtime, i dati statistici utilizzati per la valutazione delle prestazioni quali:

- Deviazione standard
- Varianza
- Percentuale di pacchetti persi

Vediamo infine alcuni test effettuati in modo da fornire una idea del percorso evolutivo del protocollo.

6.16.1 Test effettuati

Nei paragrafi successivi sono presentati alcuni test scelti tra le centinaia di prove effettuate che corrispondono ai due estremi del percorso evolutivo del protocollo, partendo dagli inizi in cui non erano utilizzati tutti gli strumenti elencati nel corso della trattazione, sino ad arrivare agli ultimi test in cui molte delle tecnologie citate sono applicate.

6.16.2 Test iniziali

Nella Figura 6.22 sono mostrati i grafici ricavati con la suite di strumenti software descritta nel paragrafo 6.14 e successivi relativi alla trasmissione di un singolo flusso tra due macchine in assenza di traffico disturbante, effettuata utilizzando una delle prime implementazioni del protocollo.

Il grafico in alto mostra i $\Delta Trj(i)$ definiti nel paragrafo 6.16 calcolati quindi sui pacchetti ricevuti nel caso di un singolo 'flusso' costituito da 10000 pacchetti inviati ad intervalli di 10ms tra due macchine connesse con un link adhoc wireless.

Nel grafico in basso invece sono mostrati i $\Delta Tsj(i)$ relativi allo stesso flusso, ovvero le differenze calcolate in corrispondenza della generazione dei pacchetti.

Osservando i dati inseriti nei grafici e relativi al receiver si vede come già con un singolo flusso di pacchetti non tutti i dati giungono a destinazione, ma alcuni vengano persi. Il valore della deviazione standard è di circa 14,5.

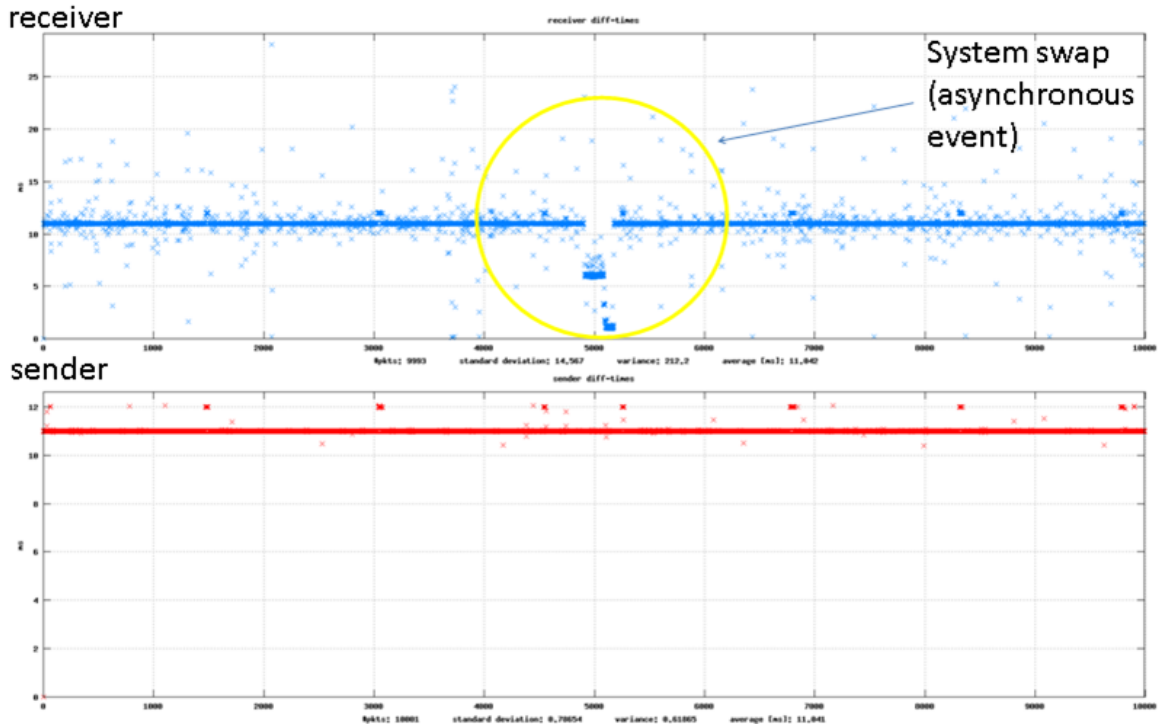


Figura 6.22: Base run test - 1 stream - no FTP

Per quanto riguarda il sender si vede come vi siano scostamenti anche di qualche millisecondo rispetto al valore teorico di invio di 10ms, ed inoltre è evidente come un fenomeno periodico influenzi l'invio ripetendosi ad intervalli abbastanza regolari. Inoltre tornando al flusso associato al receiver è presente circa in corrispondenza della metà un evento asincrono molto rilevante che interrompe la regolarità del flusso. In queste prove iniziali i massimi ritardi non sono evidenziati, ma i valori ricavati erano dell'ordine delle centinaia di millisecondi. E' da notare che i grafici in figura 6.22 sono stati ricavati in assenza di traffico FTP concorrente.

Vediamo ora in Figura 6.23 il caso in cui è invece presente traffico FTP concorrente.

E' evidente come in questo caso le prestazioni peggiorino sensibilmente nel caso del grafico relativo al receiver, con un notevole spargimento dei valori attorno al valore medio.

Si notano ancora eventi sincroni e asincroni che disturbano il flusso, il più grave dei quali è in corrispondenza della fine della sessione di comunicazione. Si vede in figura anche come i massimi siano oltre i limiti di figura. Si ricorda che i grafici fanno riferimento ad un singolo flusso tra due sole macchine.

Figura 6.23: Worst case test - 1 stream - with FTP

6.16.3 Test iniziali con kernel RT

Nella Figura 6.24 sono mostrati i grafici relativi alla trasmissione di un singolo flusso tra due macchine in assenza di traffico disturbante, effettuata in condizioni simili al paragrafo precedente, ma utilizzando una delle prime versioni della RT patch:

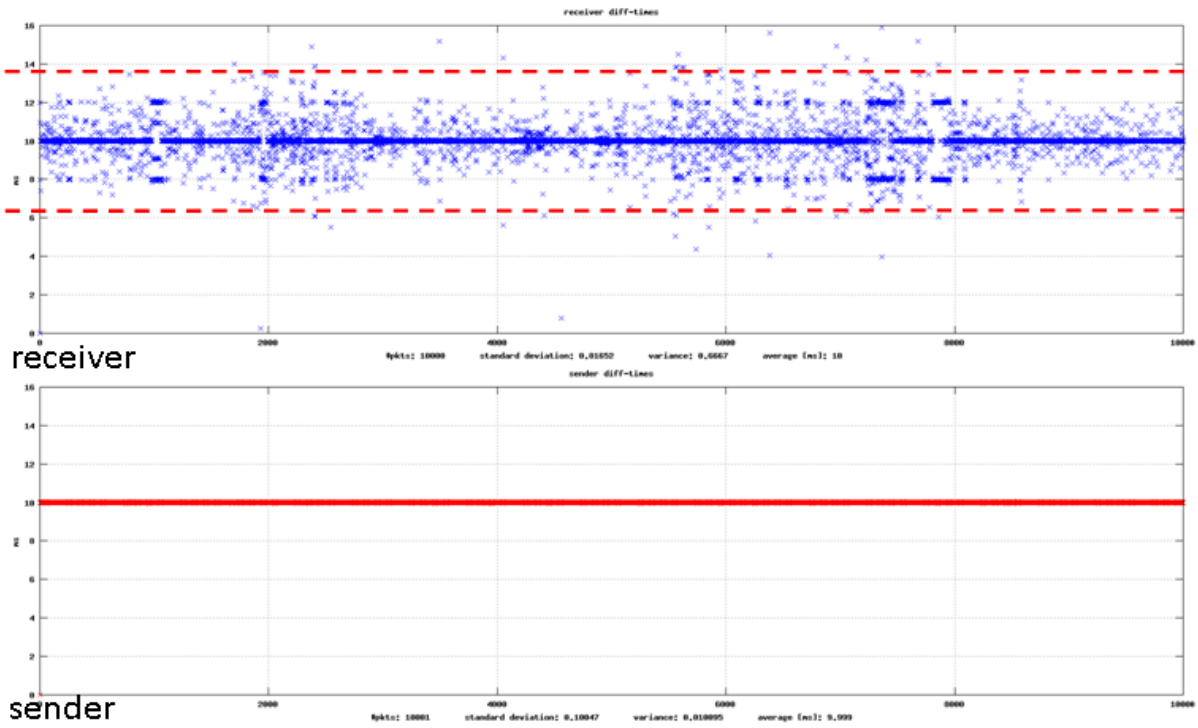


Figura 6.24: Receiver non RT Sender RT – 1 stream

Il grafico in alto mostra i $\Delta Trj(i)$ relativi al receiver che fa uso di un kernel standard. Osservando i dati inseriti nei grafici si vede come le prestazioni siano molto simili al caso precedente, a parte un piccolo miglioramento dovuto molto probabilmente al comportamento in fase di invio. Infatti in tal caso è visibile un notevole effetto del kernel con patch RT, riscontrabile nella regolarità di generazione dei pacchetti molto migliorata. I valori di $\Delta Tsj(i)$ sono infatti tutti molto vicini al valore teorico di 10ms, e così la media che ha un valore di 9,999. Anche varianza e deviazione standard sono molto ridotte di conseguenza nel caso del sender, mentre per il receiver sono simili al caso precedente.

Vediamo ora in Figura 6.25 il caso di Sender non RT e Receiver RT:

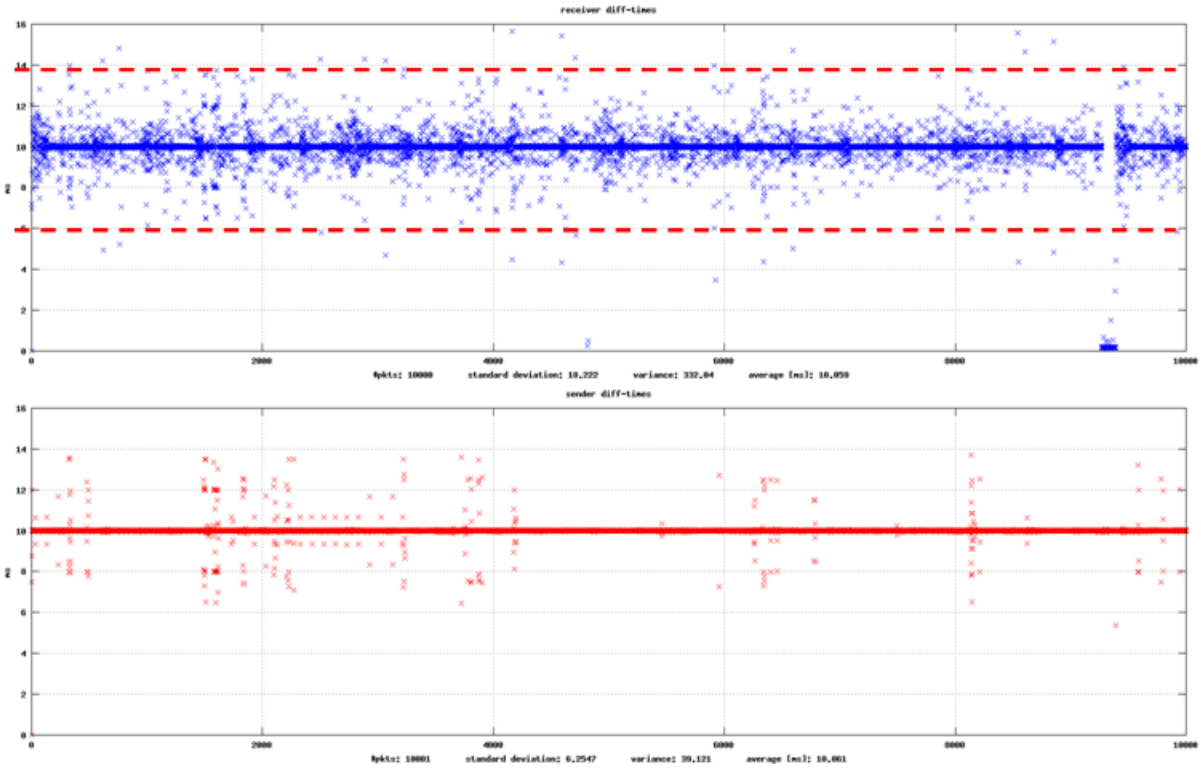


Figura 6.25: Sender non RT Receiver RT – 1 stream

E' evidente come in questo caso le prestazioni peggiorino sensibilmente nel caso del grafico relativo al sender rispetto al grafico 6.24, con un notevole spargimento dei valori attorno al valore medio. Per quanto riguarda il receiver si nota un miglioramento sopratutto in termini di eventi disturbanti, anche se si nota ancora un evento asincrono grave corrispondenza della fine della sessione di comunicazione. Ciò è probabilmente dovuto alla concomitanza di due fattori, ovvero la versione molto 'giovane' della patch RT utilizzata e la non completa ottimizzazione del software applicativo, che nella versione utilizzata in questo test ancora non sfruttava la funzionalità di lock in memoria del processo, nel il multithreading. In ogni caso è possibile apprezzare chiaramente il miglioramento di prestazioni fornito dall'utilizzo del kernel con patch RT, anche separatamente utilizzato prima nel sender e poi nel receiver.
Si ricorda che i grafici fanno riferimento ad un singolo flusso tra due sole macchine.

6.16.4 Test recenti

Nella Figura 6.26 sono mostrati invece grafici recenti relativi a prove di laboratorio effettuate trasmettendo un totale di trenta flussi con sovrapposto un trasferimento di file a banda piena. In questo caso nel test sono coinvolte quattro macchine, ovvero Imalab01-04 delle quali due

inviano flussi ad alta priorità, una invia un flusso ad alta priorità con sovrapposto il traffico FTP dovuto al trasferimento di un file di grosse dimensioni, e la quarta macchina fa da receiver. Nel test vengono utilizzate tutte le tecnologie descritte nel corso dei paragrafi precedenti:

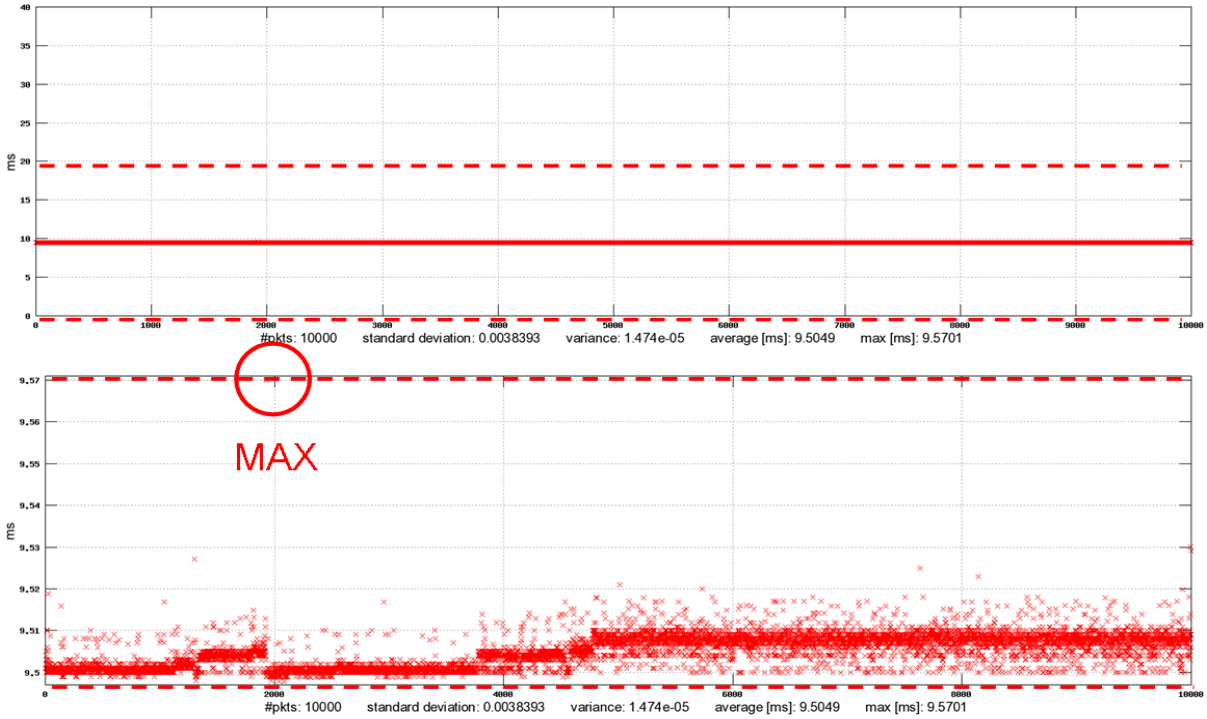


Figura 6.26: Test recenti RT + IMQ – senders
Imalab 01,03,04 vs Imalab02 30 flux + FTP full bandwidth 03->02

Il grafico in alto ed in baso sono relativi allo stesso flusso, ma con ingrandimento diverso in modo da evidenziare l'effetto ottenuto. Sono mostrati i $\Delta Tsj(i)$ relativi ad uno dei sender. Osservando i dati inseriti nei grafici si vede come le prestazioni sono notevolissimamente migliorate rispetto ai casi precedenti. Non vi è nessun pacchetto perso, nonostante la concorrenza di trenta flussi simili ai casi analizzati in precedenza invece di uno solo, e la presenza in contemporanea di un flusso FTP a banda piena, ed i valori sono talmente regolari da sembrare una linea retta. La regolarità di generazione dei pacchetti è notevolissima, vicino al valore medio. Dal grafico in basso che ha una scala molto ingrandita (è stato generato con autoscaling attivo parametrato in modo da scegliere come minimo e massimo valori pesati con la deviazione standard) si vede che i valori di $\Delta Tsj(i)$ sono compresi tra 9,49ms e 9,57ms dove si situa il massimo evidenziato che risulta tra l'altro anche sparso. Il valore medio si situa attorno a 9,5ms in quanto lo scheduler del kernel con patch RT configurato ha risoluzione di circa 1ms, per cui il processo sender viene eseguito sempre a circa 9,5ms di intervallo. La

deviazione standard ha un valore molto basso, con due zeri dopo la virgola. Tale situazione si può definire praticamente ideale nel caso del sender, considerando che il limite massimo di ritardo considerato da specifica ISO è dell'ordine dei 100ms.

Vediamo ora in Figura 6.27 cosa accade in ricezione:

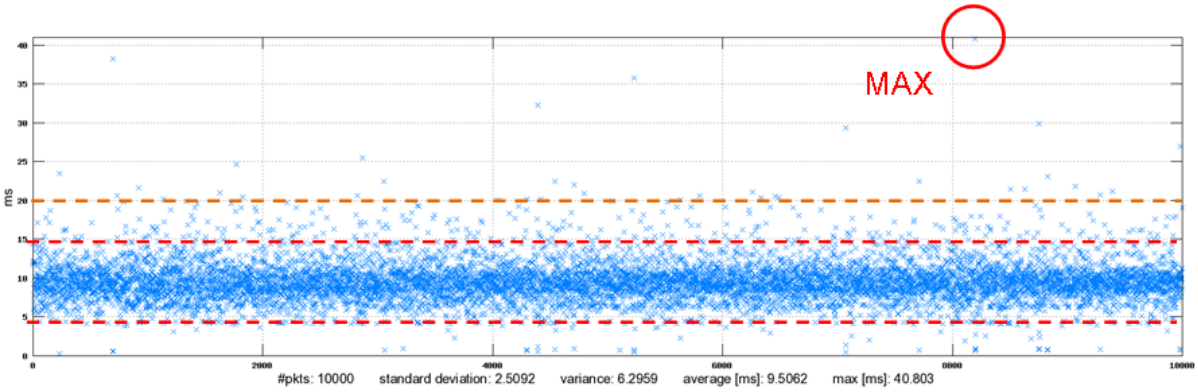


Figura 6.27: Test recenti RT + IMQ – receivers
Imalab 01,03,04 vs Imalab02 30 flux + FTP full bandwidth 03->02

E' evidente come anche in questo caso le prestazioni siano notevolissimanente migliorate, considerando la differenza di situazione. Il grafico è stato generato con la funzione di autoscaling attivata. Si può vedere come i valori siano per una amplissima maggioranza compresi tra 5ms e 15ms, con un numero minimo oltre 20ms ed un massimo a 40ms (target 100ms). Oltre a ciò tutti i pacchetti sono stati ricevuti anche in condizioni di notevole affollamento della rete. Deviazione standard vale circa 2,5 valore decisamente migliorativo rispetto ai casi valutati in precedenza.

Non si nota più nè nel sender nè nel receiver alcun evento disturbante, ma anzi la dispersione attorno al valor medio è decisamente regolare, cosa che è indice di una buona regolarità di funzionamento del sistema.

Si ricorda che i grafici fanno riferimento in questo caso a trenta flussi ad alta priorità con sovrapposto un trasferimento FTP a banda piena in una rete adhoc di quattro macchine.

Nel seguito sono inseriti gli estratti dei file CSV contenenti dati statistici dei flussi dai quali sono stati ricavati i grafici in questo paragrafo, in modo da dare una idea quantitativa oltre che qualitativa per via grafica dei valori ottenuti.

Nella seguente Figura 6.28 sono mostrati i dati associati ad alcuni dei flussi estratti tra quelli da cui sono stati ricavati i grafici di Figura 6.26:

```
filename,standard deviation,variance,average,max,exp packet number,packet number
p50000.dat,0.006619,0.000044,9.504256,9.689941,10000,10000
p50001.dat,0.008461,0.000072,9.502717,9.534912,10000,10000
p50002.dat,0.006612,0.000044,9.504010,9.679932,10000,10000
 - - - - -
p50010.dat,0.004864,0.000024,9.501719,9.731934,10000,9878
p50011.dat,0.004969,0.000025,9.501336,9.746094,10000,9885
p50012.dat,0.004516,0.000020,9.501415,9.581055,10000,9909
 - - - - -
p50020.dat,0.005524,0.000031,9.500532,9.989990,10000,10000
p50021.dat,0.002419,0.000006,9.500785,9.532959,10000,10000
p50022.dat,0.004005,0.000016,9.499956,9.583008,10000,10000
 - - - - -
```

Figura 6.28: Test recenti RT + IMQ – senders stats
Imalab 01,03,04 vs Imalab02 30 flux + FTP full bandwidth 03->02

Nella seguente Figura 6.29 sono mostrati i dati associati ad alcuni dei flussi estratti tra quelli da cui sono stati ricavati i grafici di Figura 6.27:

```
filename,standard deviation,variance,average,max,exp packet number,packet number
p50000.dat,2.793287,7.802453,9.506008,48.388916,10000,10000
p50001.dat,2.752244,7.574847,9.503070,51.100098,10000,10000
p50002.dat,2.947219,8.686098,9.505777,43.335938,10000,10000
 - - - - - -
p50010.dat,3.315288,10.991135,9.508772,52.658936,10000,9878
p50011.dat,3.321477,11.032210,9.506202,53.856934,10000,9885
p50012.dat,3.344257,11.184053,9.511052,54.524902,10000,9909
 - - - - - -
p50020.dat,3.132272,9.811126,9.503961,45.917969,10000,10000
p50021.dat,2.529328,6.397502,9.502254,52.454102,10000,10000
p50022.dat,3.016240,9.097706,9.501735,45.910889,10000,10000
 - - - - - -
```

Figura 6.29: Test recenti RT + IMQ – receivers stats
Imalab 01,03,04 vs Imalab02 30 flux + FTP full bandwidth 03->02

Nelle figure precedenti sono evidenziati separatamente alcuni flussi provenienti dalle diverse macchine coinvolte nel test. In particolare i flussi da 0 a 9 provengono da Imalab01, quelli da 10 a 19 da Imalab03 e quelli da 20 a 29 da Imalab04. Imalab02 è il receiver.

Dalla osservazione dei dati nelle due figure precedenti si può osservare come in tutti i flussi trasmessi dalle macchine non impegnate nel trasferimento FTP vi siano zero pacchetti persi, mentre per la sola macchina impegnata nel trasferimento file (Imalab03) risultano alcuni pacchetti persi. Ciò è in accordo con la teoria e dimostra il buon funzionamento della patch IMQ, senza la quale la perdita di pacchetti si rivela su tutti i flussi, rimanendo maggiore nel caso di Imalab03.

6.17 Discussione dei test

Nonostante il numero di test presentato sia notevolmente riduttivo della mole totale prodotta, l'osservazione degli stessi permette di ricavare alcune considerazioni di seguito sinteticamente elencate.

- Le strategie di priorità e architetturali utilizzate consentono rilevanti prestazioni real-time anche con utilizzo di piattaforme non appositamente sviluppate
- Notevole effetto del kernel con patch RT – anche se 'giovane'
- Effetto significativo patch IMQ recenti versioni – sopratutto minore perdita di pacchetti
- Test effettuati con sino a 45 flussi non mostrano significativi decadimenti di prestazioni, anche in presenza di concomitante trasferimento dati che tende a saturare la banda
- Architettura software ancora non completamente ottimizzata (tuning kernel RT, tuning scheduler, tuning IMQ e nuove versioni)
- Supporto 802.11n reti adhoc immaturo (molto importante prestazioni teoriche decuplicate)
- Software completamente in userspace da spostare in kernel mode (kernel daemon, kernel patch o modulo kernel)

6.18 Task 2: Conclusioni

Il lavoro svolto nel corso del presente dottorato in riferimento al Task 2 dimostra che il problema di progettare u protocollo isocrono real-time sembra risolto almeno dal punto di vista delle prestazioni richieste per l'invio di pacchetti, in relazione alle specifiche richieste dalle applicazioni attualmente prese come riferimento. Le prestazioni con 5 macchine in rete e fino a 45 flussi mostrano che un sistema quale quello implementato può soddisfare le specifiche richieste da una applicazione nel mondo reale. Dal punto di vista della ricezione le prestazioni raggiunte sono già buone, e sono considerevolmente migliorate con l'andare avanti

dello sviluppo. In ogni caso ci sono svariati miglioramenti che possono derivare da vari fattori: ad esempio l'utilizzo dello standard 802.11n, che dovrebbe avere prestazioni notevolmente superiori rispetto (di un fattore 10 per certi parametri) ad 802.11g utilizzato nel corso del presente lavoro, proprio nelle aree più rilevanti, ovvero la gestione dei conflitti di accesso al mezzo trasmissivo e la banda. La gestione del traffico in ingresso realizzata tramite la patch IMQ ha già migliorato le prestazioni, sopratutto nelle ultime versioni della patch che hanno fornito risultati superiori rispetto alle precedenti. Oltre a ciò la patch 'Agro Kernel' è in fase di test, e potrebbe contribuire a migliorare ulteriormente le prestazioni. Ancora è probabile che miglioramenti implementativi in termini di architettura software, come ad esempio il passaggio in kernel mode di parte del codice, o la ottimizzazione ulteriore del kernel con patch RT o IMQ o la più accurata messa a punto del sistema possano fornire ancora significativi incrementi di prestazioni.

Tutte queste possibilità lasciano ben sperare che il lavoro svolto nel presente Dottorato possa portare ad una proposta di nuovo Standard Internazionale che vedrà un concreto futuro.

Intanto tutti gli scopi previsti sono stati raggiunti, ed è stata progettata ed implementata una serie di strumenti, un ambiente di sviluppo e vari siti di test, e ancora piattaforme hardware e software ed un protocollo di massima che consentono di proseguire il lavoro oltre i confini del presente studio, ed hanno sinora contribuito ad ottenere gli importanti risultati descritti in dettaglio nel Capitolo 2.

6.19 Task 2: Sviluppi futuri

Di seguito è presente un sintetico elenco di nuove prospettive per lo sviluppo del protocollo:

- Implementazione stabile di Inbound Packet priority nel Linux Kernel (Agro Kernel o utilizzo IMQ se prestazioni elevate e integrato nel kernel mainstream)
- Messa a punto di componenti del sistema con buone prestazioni ma ancora in forte evoluzione, ad esempio patch RT e IMQ kernel
- Utilizzo del nuovo protocollo 802.11n
- Valutazione approfondita di piattaforme embedded maggiormente orientate verso real time
- Valutazione con priorità RT maggiori di 89, ovvero processi di sistema
- Ottimizzazione dei kernel con riconfigurazione mirata e ricompilazione
- Ottimizzazione del kernel RT tramite strumenti di configurazione

- Ottimizzazione architettura software del protocollo, con inserimento in kernel space (vedi roadmap in Appendice C, Figura C.2
- Definizione della architettura definitiva del protocollo in termini di prestazioni, di architettura complessiva, di servizi offerti (dipende da scelte ISO ancora da effettuarsi)
- Valutazione di un modello di protocollo che fornisca o meno gestione/arbitraggio del traffico e servizi avanzati (metaconnessioni) - vedi punto precedente
- Valutazione di varie strategie di ritrasmissione - vedi punti precedenti

Capitolo 7

7.1 Pubblicazioni e prodotti

Articolo pubblicato: vedi appendice articoli

Alfredo Revenaz, Massimiliano Ruggeri, Massimo Martelli, Wireless Communication Protocol for Agricultural Machines Synchronization and Fleet Management, International Symposium on Industrial Electronics (ISIE) 2010 Bari. ISIE2010 proceedings p.3498-3504 ISBN/ISSN 978-1-4244-6391-6.

Articolo accettato: sarà presentato a Giugno 2011 - vedi appendice articoli

Alfredo Revenaz, Massimiliano Ruggeri, Velio Tralli, Low Latency Wi-Fi Realtime Protocol for Agricultural Machines Synchronization Using Linux RT Kernel, International Symposium on Industrial Electronics (ISIE) 2011 Gdansk, Poland.

Articolo in preparazione:

Per sottomissione a IEEE Communications Magazine. E.T.A. Estate 2011

Nuova normativa ISO in preparazione: vedi Capitolo 2 paragrafo 2.2.12

Tractors and machinery for agriculture and forestry — Wireless networks — Part 1: General

7.2 Conclusioni finali

Obiettivi diversi e dinamici hanno caratterizzato il lavoro svolto durante il presente Dottorato, che ha visto una varietà e vastità di argomenti coinvolti. Risultati differenti sono stati ottenuti per i tre Task in cui è stato logicamente diviso il lavoro, che ha visto concludersi in una fase preliminare i primi due, e proseguire il terzo sino al raggiungimento di un importante risultato. Il percorso effettuato è stato in realtà condizionato dal rapporto con ISO, che dopo una fase iniziale di valutazione ha di fatto delineato le priorità del lavoro da svolgere, manifestando la maggiore urgenza del raggiungimento di uno Standard Internazionale per la comunicazione short range, che ha dato poi origine al lavoro svolto nel Task 2.

In ogni caso tutti e tre i Task hanno raggiunto una fase di sviluppo che si può dire compiuta.

Per quanto riguarda il Task 0 è stato completamente progettato un sistema standard di gestione flotte e dati di produzione da utilizzarsi in campo agricolo, e la parte realizzata come Proof of Concept è un vero e proprio sistema funzionante completo di database su RDBMS e software di gestione in grado di interagire con uno (o più) terminali GPS memorizzandone i dati.

La parte realizzata inoltre dimostra compiutamente le possibilità descritte nel progetto iniziale, avendo implementato tutti i concetti cardine in un prototipo funzionante, e che può essere preso a riferimento per sviluppi futuri che portino a formulare una proposta compiuta di Standard Internazionale.

Le finalità del Task 0 sono state dunque pienamente raggiunte.

Per quanto riguarda il Task 1 le attività svolte nel presente dottorato hanno compreso lo studio del complesso Standard ISOBUS attuale che prevede una comunicazione solo intraveicolare, che è stato messo poi concettualmente in relazione con lo scenario prospettato nel Task 0.

In seguito a tale analisi è stato realizzato il progetto di massima di una soluzione che permetterebbe di implementare una comunicazione sicura nell'ambito del sistema complessivo costituito da un network ISOBUS e da un sistema di gestione dati remoto raggiunto via Internet. La soluzione prospettata ha numerosi e rilevanti vantaggi, ed è attualmente in fase di valutazione. Le attività riferite al Task 1 sono terminate dopo la realizzazione del progetto di massima descritto, e si è proceduto con le attività legate al Task 2. Il lavoro svolto in riferimento a quest'ultimo Task è stato progettare e realizzare almeno in versione preliminare un protocollo di comunicazione wireless dedicato al fleet management e alla sincronizzazioni di veicoli ed accessori nel campo delle macchine agricole, che è oggetto di proposta per un nuovo Standard Internazionale ISO.

Le attività svolte ed i risultati raggiunti lasciano ben sperare che il lavoro svolto nel presente Dottorato possa portare ad una proposta che vedrà un concreto futuro.

Intanto durante le varie fasi del lavoro svolto è stata progettata ed implementata una serie di strumenti, un ambiente di sviluppo e vari siti di test, e ancora piattaforme hardware e software ed un protocollo in versione preliminare che consentono di proseguire il lavoro oltre i confini del presente studio, ed hanno sinora contribuito ad ottenere gli importanti risultati descritti in dettaglio nel Capitolo 2.

Si può dire dunque che pure nella loro dinamicità tutti gli obiettivi definiti nel corso del presente Dottorato sono stati raggiunti.

Ringraziamenti

Questa tesi è stata svolta in gran parte con la collaborazione dell'istituto CNR-IMAMOTER (Istituto per le macchine agricole e movimento terra) di Ferrara. Desidero pertanto ringraziare tutto il personale dell'istituto per aver messo a disposizione gli strumenti hardware e software necessari e per la cordialità mostratami.

Un ringraziamento al Prof. Velio Tralli, Tutore interno, per i suoi utili consigli, la sua disponibilità e collaborazione.

Un ringraziamento al Prof. Massimiliano Ruggeri, Tutore esterno, per la disponibilità, la collaborazione e l'attenzione dedicata nello svolgimento del lavoro e per la realizzazione della tesi.

Riferimenti

Norme Internazionali

1. SAE J 1939 (all parts), SAE International, U.S.A., 2006.
2. ISO 11783 (all parts) Tractors and machinery for agriculture and forestry — Serial control and communications data network, ISO Organization, Geneva, Switzerland, 2009.
3. ISO 13849-1 (Safety of machinery, Safety-related parts of control systems, Part 1: General principles for design), ISO Organization, Geneva, Switzerland, 2006.
4. IEC 61508, Functional safety of electrical/electronic/programmable electronic safety-related systems – Parts 1-7, IEC Organization, Geneva, Switzerland, 2007.
5. ISO 15998, Earth-moving machinery — Machine-control systems (MCS) using electronic components — Performance criteria and tests for functional safety, ISO Organization, Geneva, Switzerland, 2009.
6. ISO 15998, MWMS Machine Work Management System - Safety on Earth Moving Machines ISO Organization, Geneva, Switzerland, 2008
7. ISO 25119 (all parts) Tractors and machinery for agriculture and forestry -- Safety-related parts of control systems, ISO Organization, Geneva, Switzerland, 2010.

Articoli

8. Alfredo Revenaz, Massimiliano Ruggeri, Massimo Martelli, Wireless Communication Protocol for Agricultural Machines Synchronization and Fleet Management, 2010, International Symposium on Industrial Electronics (ISIE 2010), July 2010
9. H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, The Internet Society, July 2003.
10. C. Mbarushimana, A. Shahrabi, “Comparative Study of Reactive and Proactive Routing Protocols Performance in Mobile Ad Hoc Networks”, 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07), 2007
11. Fu Yongsheng , Wang Xinyu , Li Shanping, “Performance Comparison and Analysis of Routing Strategies in Mobile Ad Hoc Networks”, 2008 International Conference on Computer Science and Software Engineering, December 2008.

12. Glaucia Campos , Gledson Elias, “Performance Issues of Ad Hoc Routing Protocols in a Network Scenario used for Videophone Applications”, Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 9 , 2005.

13. Toby Xu , Ying Cai, Streaming in MANET: Proactive Link Protection and Receiver-Oriented Adaptation, 2007 IEEE International Performance, Computing, and Communications Conference , April 2007.

14. Vun, N.; Hor, H.F.; Chao, J.W.; , "Real-Time Enhancements for Embedded Linux," Parallel and Distributed Systems, 2008. ICPADS '08. 14th IEEE International Conference on , vol., no., pp.737-740, 8-10 Dec. 2008 doi: 10.1109/ICPADS.2008.108

15. Betz, W.; Cereia, M.; Bertolotti, I.C.; , "Experimental evaluation of the Linux RT Patch for real-time applications," Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on , vol., no., pp.1-4, 22-25 Sept. 2009 doi: 10.1109/ETFA.2009.5347056

16. Mitsching, R.; Weise, C.; Gatterdam, T.; Kowalewski, S.; , "Low Effort Evaluation of Real-Time and Reliability Requirements for Embedded Systems," Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on , vol., no., pp.2433-2440, June 29 2010-July 1 2010 doi: 10.1109/CIT.2010.418

17. Marchesin, A.; , "Using Linux for real-time applications," Software, IEEE , vol.21, no.5, pp. 18- 20, Sept.-Oct. 2004 doi: 10.1109/MS.2004.1331295

18. Zhu Xiangbin; , "Support QoS in Open Real-Time Systems," Computer Science and Software Engineering, 2008 International Conference on , vol.2, no., pp.190-193, 12-14 Dec. 2008 doi: 10.1109/CSSE.2008.827

19. Bao Rong Chang; Chung-Ping Young; Hsiu Fen Tsai; Ren-Yang Fang; , "Embedded System for Inter-Vehicle Heterogeneous Wireless-Based Real-Time Multimedia Streaming and Video/Voice over IP," Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on , vol., no., pp.365-368, 7-9 Dec. 2009 doi: 10.1109/ICICIC.2009.192

20. Song Kai; Yan Liping; , "Improvement of Real-Time Performance of Linux 2.6 Kernel for Embedded Application," Computer Science-Technology and Applications, 2009. IFCSTA '09. International Forum on , vol.2, no., pp.71-74, 25-27 Dec. 2009 doi: 10.1109/IFCSTA.2009.138

21. Vun, N.; Hor, H.F.; Chao, J.W.; , "Real-Time Enhancements for Embedded Linux," Parallel and Distributed Systems, 2008. ICPADS '08. 14th IEEE International Conference on , vol., no., pp.737-740, 8-10 Dec. 2008 doi: 10.1109/ICPADS.2008.108

22. Rui He; Man Yuan; Jianping Hu; Hong Zhang; Zhigang kan; Jian Ma; , "A real-time scalable and dynamical test system for MANET," Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003. 14th IEEE Proceedings on , vol.2, no., pp. 1644-1648 vol.2, 7-10 Sept. 2003 doi: 10.1109/PIMRC.2003.1260393

23. Sojka, M.; Pi?s?a, P.; Petera, M.; S?pinka, O.; Hanzalek, Z.; , "A comparison of Linux CAN drivers and their applications," Industrial Embedded Systems (SIES), 2010 International Symposium on , vol., no., pp.18-27, 7-9 July 2010 doi: 10.1109/SIES.2010.5551367

24. Potlapally, N.R.; Ravi, S.; Raghunathan, A.; Jha, N.K.; , "A study of the energy consumption characteristics of cryptographic algorithms and security protocols," Mobile Computing, IEEE Transactions on , vol.5, no.2, pp. 128- 143, Feb. 2006 doi: 10.1109/TMC.2006.16

25. Cooling, J.; , "Real-time operating systems for the embedded world," Open Control Systems - The Importance of Industrial Standards , vol., no., pp. 4/0- 410, 26 May 2004 doi: 10.1049/ic:20040128

Testi

26. Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Riccardo Torlone, Basi di dati: modelli e linguaggi di interrogazione, Milano, Mc-Graw Hill, 2002

27. Alessandro Bellini, Andrea Guidi, Linguaggio C: guida alla programmazione, Milano, Mc-Graw Hill, 2006

28. Stephen John Cannan, Gerard Otten, Il manuale SQL, Milano, Mc-Graw Hill, 1994

29. Philippe Dax, Programmare in linguaggio C, Milano, F. Angeli, 1985

30. Brian W. Kernighan, Dennis M. Ritchie, Linguaggio C, Milano, Jackson, 1989

31. Richard W. Stevens, UNIX Network Programming, Volume 1: The Sockets Networking API, Third Edition, Addison-Wesley Professional, 2006

32. Richard W. Stevens, UNIX Network Programming, Volume 2: Interprocess Communications: Second Edition, Prentice Hall PTR, 2007

33. Richard W. Stevens, Stephen A. Rago, Advanced Programming in the UNIX Environment: Second edition, Addison-Wesley Professional, 2008

34. Richard W. Stevens, TCP/IP Illustrated, Vol. 1: The Protocols, Addison-Wesley Professional Computing Series, 1994)

35. Richard W. Stevens, Gary R. Wright, TCP/IP Illustrated, Vol. 2: The Implementation, Addison-Wesley Professional, 1995)

36. Richard W. Stevens, TCP/IP Illustrated, Vol. 3: TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols, Addison-Wesley Professional, 1996)

37. Bruce Schneier, Applied Cryptography, Protocols, Algorithms, And Source Code In C: 2Nd Edition, John Wiley & Sons, 1996

38. Baldoni, Ciliberto, Cattaneo, Elementary number theory, cryptography and codes, Springer, 2009

39. Stallings, William, Cryptography And Network Security: 4Th Ed, Prentice Hall, 2005

In Internet

40. Lista delle Commissioni Tecniche ISO. . . . http://www.iso.org/iso/standards_development

41. Free and open source software . . . . . .

http://en.wikipedia.org/wiki/Free_and_open_source_software

42. Free software licence . . . . . http://en.wikipedia.org/wiki/Free_software_licence

43. PostgreSQL RDBMS . . . . . .http://www.postgresql.org/

44. PostgreSQL Documentation . . . . . .http://www.postgresql.org/docs/

45. National Marine Electronics Association - NMEA . . . . . .http://www.nmea.org/

46. NMEA 0183 specification . . . . . http://en.wikipedia.org/wiki/NMEA_0183

47. PC104 Embedded Consortium http://www.pc104.org/

48. PC104 Consortium - specifiche degli standard. . .http://www.pc104.org/specifications.php

49. Gpsd . . . . . .http://gpsd.berlios.de/

50. JavaScript Object Notation . . . . . .http://en.wikipedia.org/wiki/JSON

51. Data Encryption Standard . . . . . .http://en.wikipedia.org/wiki/Data_Encryption_Standard

52. Advanced Encryption Standard . . . . . .

http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

53. RSA Encryption Standard. . . . . http://en.wikipedia.org/wiki/RSA

54. Rabin Cryptosystem . . . . . http://en.wikipedia.org/wiki/Rabin_cryptosystem

55. ElGamal Encryption . . . . . .http://en.wikipedia.org/wiki/ElGamal_encryption

56. Norma ISO 11783-3:2007. . . . . .

http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=41160

57. Norma ISO 11783-6:2010. . . . . .

http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=42725

58. LynuxWorks Inc. - BlueCat Linux operating system . . . . . .http://www.lynuxworks.com/

59. MontaVista Software, Real-time Linux . . . . . http://www.mvista.com/

60. Politecnico di Milano, Dip. di Ingegneria Aerospaziale. RTAI . . . . . .https://www.rtai.org/

61. RedHat Enterprise MRG Realtime . . . . . http://www.redhat.com/mrg/realtime

62. Real-Time Linux Wiki . . . . . .https://rt.wiki.kernel.org

63. A realtime preemption overview http://lwn.net/Articles/146861

64. The Community ENTerprise OS - CentOS Linux . . . . . http://www.centos.org/

65. Scientific Linux by Fermilab, CERN . . . . . .http://www.scientificlinux.org/

66. Linux TC Howto. . . . . http://linux-ip.net/articles/Traffic-Control-HOWTO/

67. Linux TC Next Generation . . . . . .http://tcng.sourceforge.net/

68. Linux Intermediate Queueing Device (IMQ) . . . . . .http://www.linuximq.net

69. Linux IMQ Wiki . . . . . http://wiki.nix.hu/cgi-bin/twiki/view/IMQ/WebHome

70. Apache Subversion . . . . . http://subversion.apache.org/

71. GNU Octave . . . . . .http://www.gnu.org/software/octave/

72. IMAMOTER - CNR . . . . . http://www.imamoter.cnr.it/

Appendice A

Schema Mobile Information System (MI System)

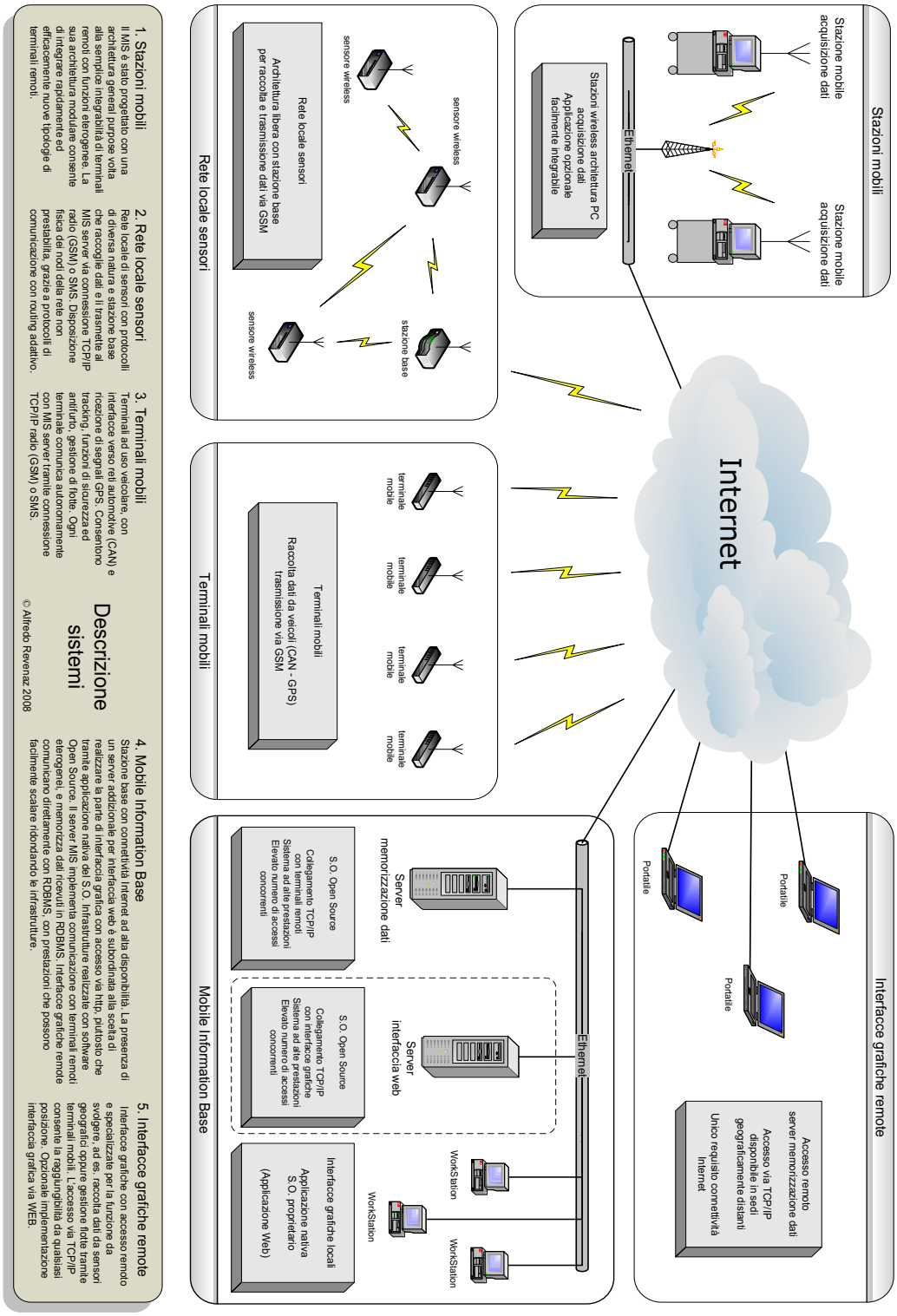


Figura A.1: Schema generale Mobile Information System

Schema Mobile Information Server (MI Server)

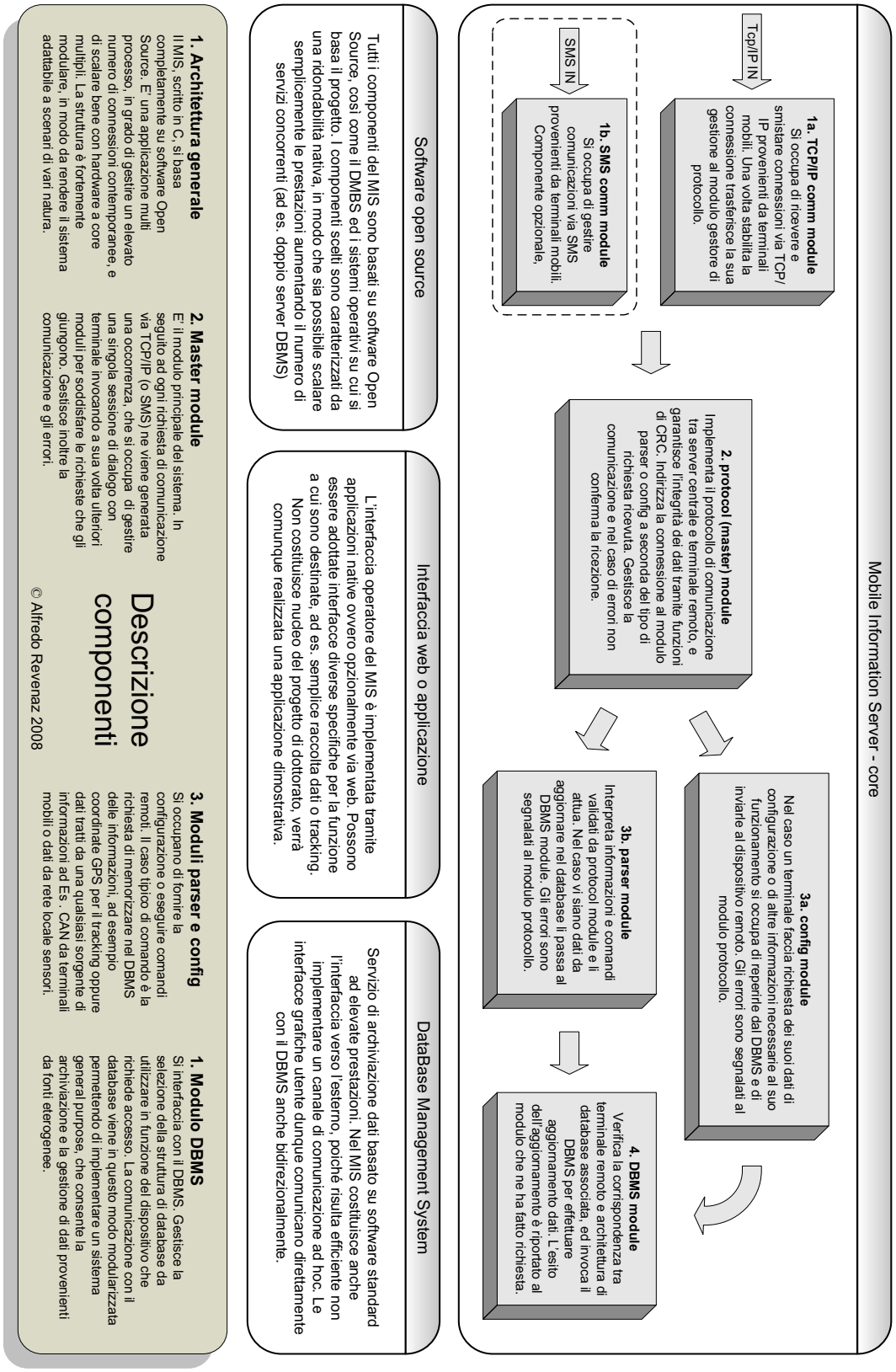


Figura A.2: Schema Mobile Information Server

Appendice B

Struttura del Database

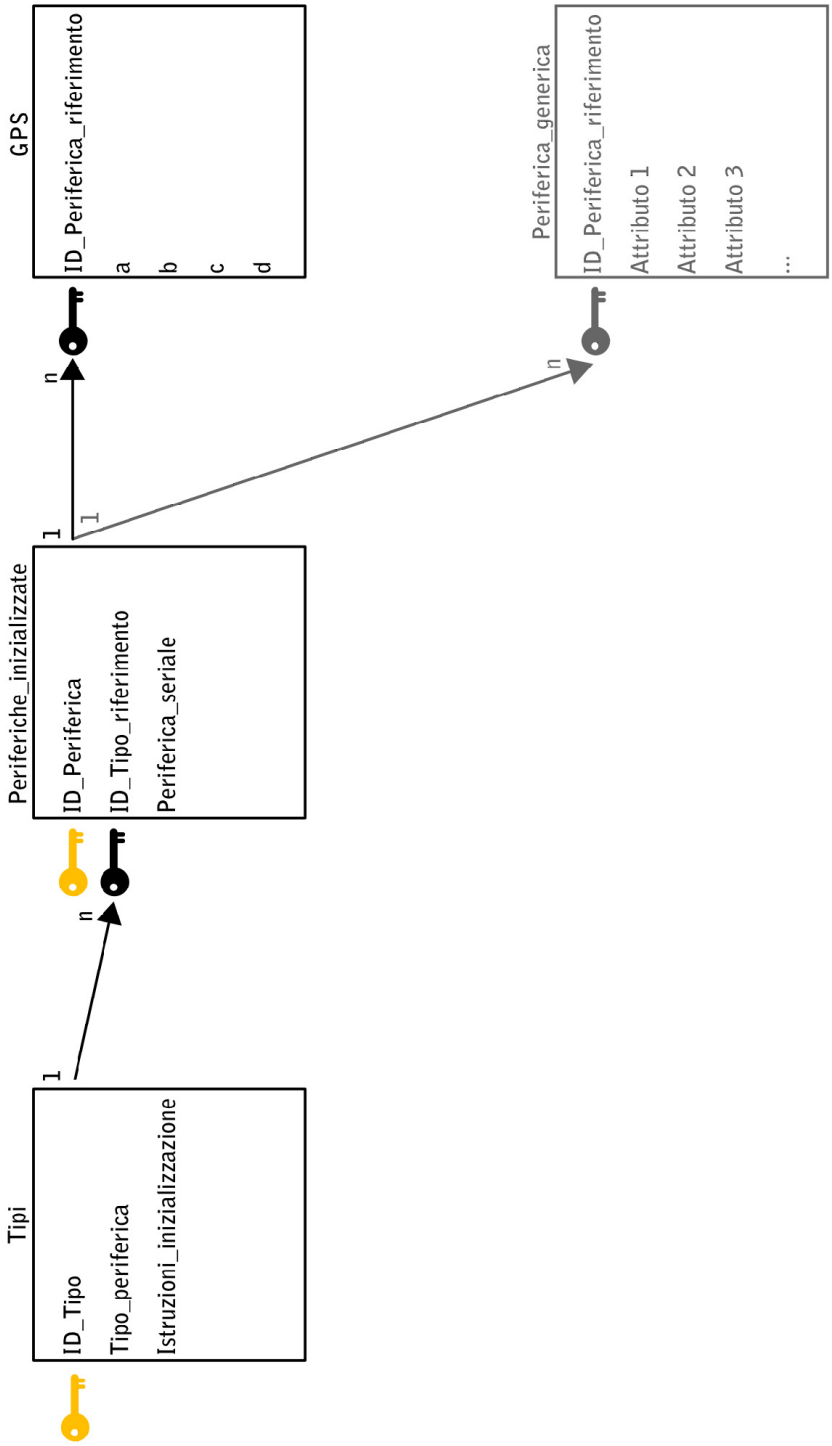


Figura B.1: Struttura del Database nel Task 0

Appendice C

Kernel packet traveling diagram

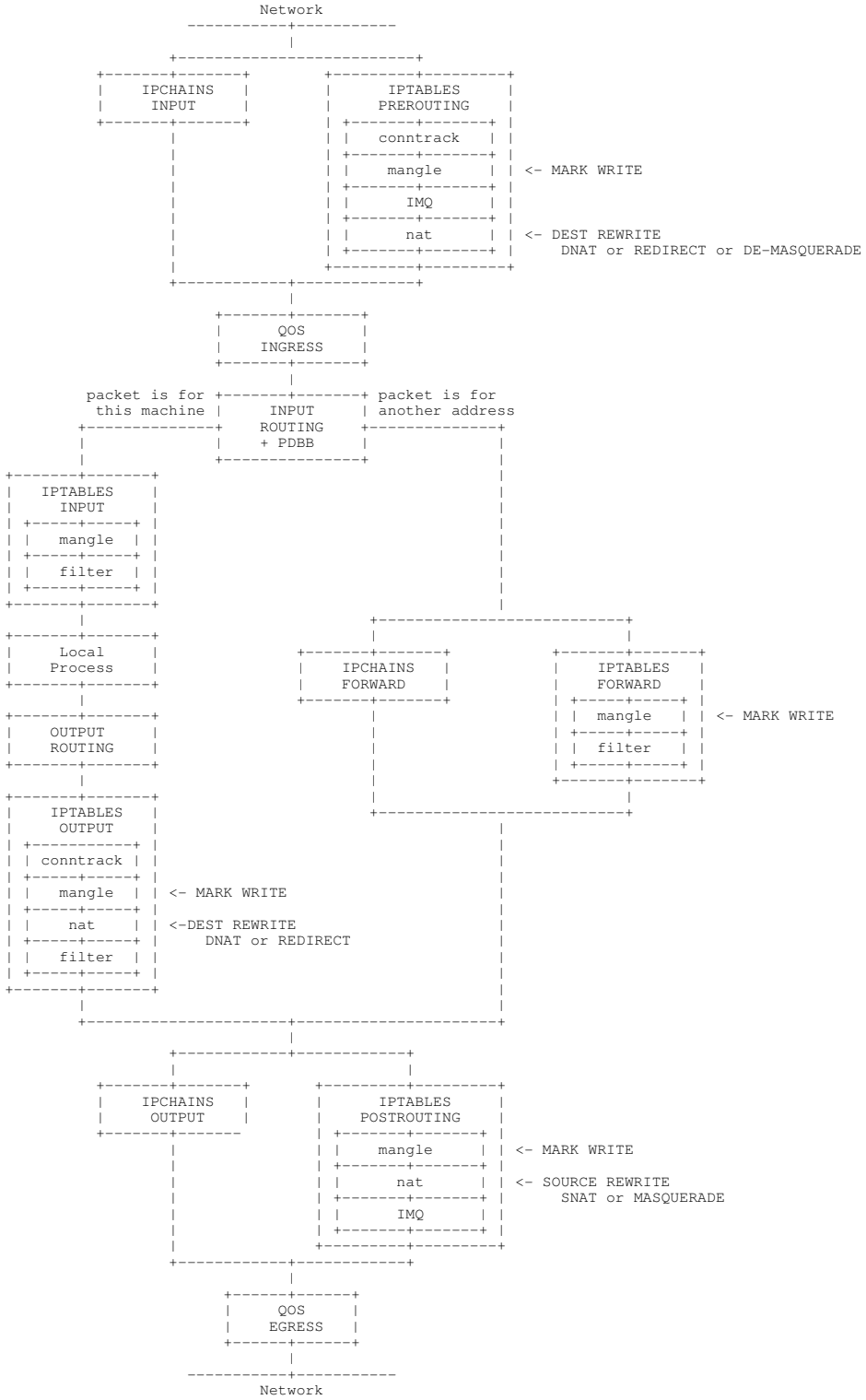


Figura C.1: Linux kernel packet traveling diagram

Roadmap implementazione AEMCP protocol

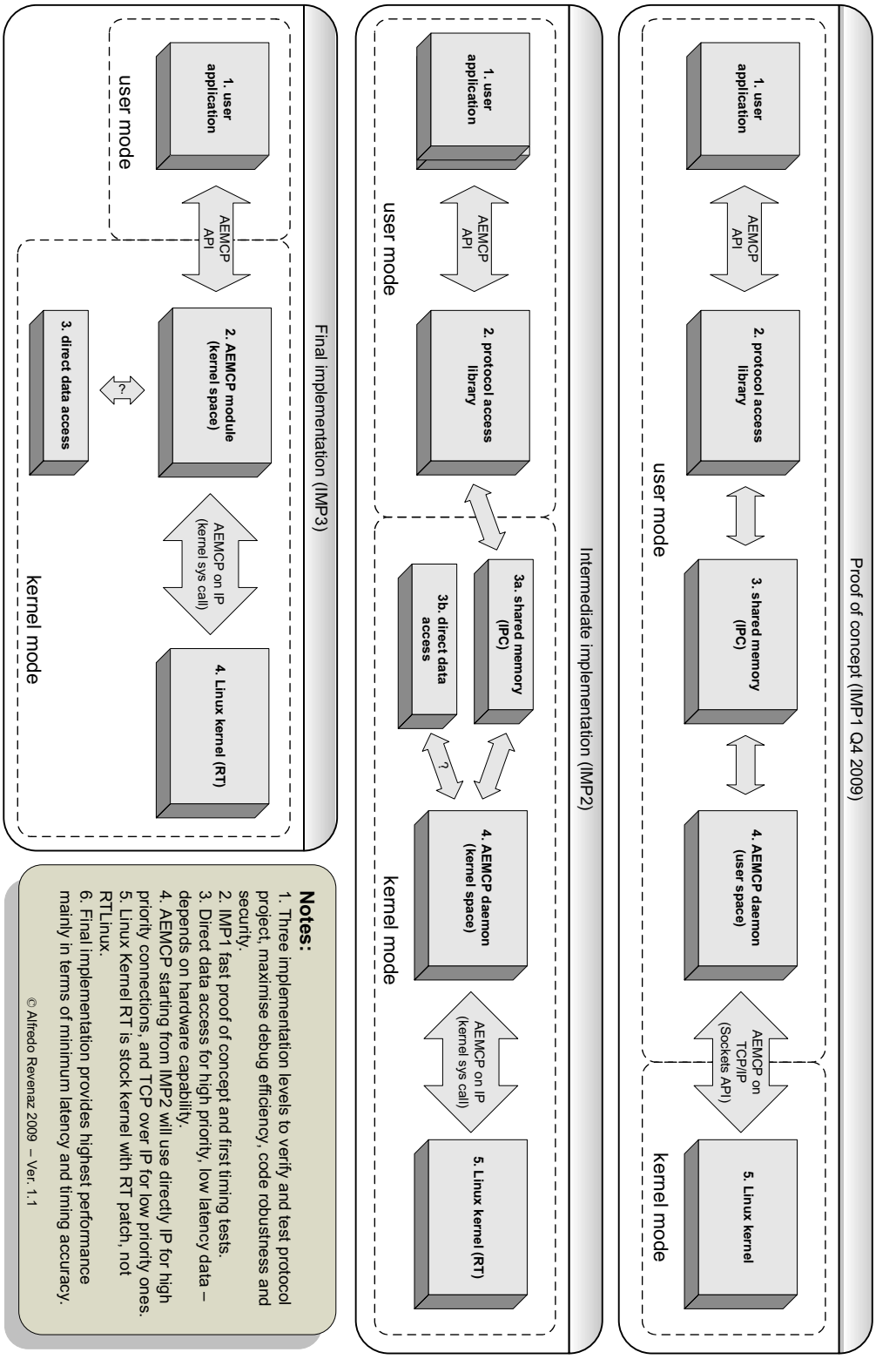


Figura C.2:Roadmap AEMCP protocol

Field test sheet

Test sheet

Date:		Number:	

Test run	Prio rec task	Prio send task	Prio UDP traffic	FTP traffic	Time	Comments
0	no	no	no	no		base run - no options
1	no	no	no	si		no options + FTP traffic
2	no	no	si	no		only prio UDP traffic
3	no	no	si	si		prio UDP traffic + FTP traffic
4	no	si	no	no		prio send task
5	no	si	no	si		prio send task + FTP traffic
6	no	si	si	no		prio send task + prio UDP traffic
7	no	si	si	si		prio send task + prio UDP + FTP traffic
8	si	no	no	no		prio rec task
9	si	no	no	si		prio rec task + FTP traffic
10	si	no	si	no		prio rec task + prio UDP traffic
11	si	no	si	si		prio rec task + prio UDP traffic + FTP traffic
12	si	si	no	no		prio send and rec task
13	si	si	no	si		prio send and rec task + FTP traffic
14	si	si	si	no		prio send and rec task + prio UDP traffic
15	si	si	si	si		prio send and rec task + prio UDP traffic + FTP traffic

Figura C.3: Field test sheet

Appendice D

Glossario

3DES . . . . . Triple Data Encryption Standard

ACPI . . . . . Advanced Configuration and Power Interface

AES . . . . . Advanced Encryption Standard

APM . . . . . Advanced Power Management

DES . . . . . Data Encryption Standard

DBMS . . . . . Data Base Management System

FLOSS . . . . . Free Libre and Open Source Software

GPS . . . . . Global Positioning System

GPSD . . . . . GPS Daemon

ICT . . . . . Information and Communication Technologies

IDE . . . . . Integrated Development Environment

ISO . . . . . International Organization for Standardization

JSON . . . . . JavaScript Object Notation

NMEA . . . . . National Marine Electronics Association

O.S. . . . . Operating System

OSI . . . . . Open Systems Interconnection

RDBMS . . . . . Relational Data Base Management System

S.O. . . . . Sistema Operativo

TCP . . . . . Transmission Control Protocol

Articoli

1. International Symposium on Industrial Electronics (ISIE) 2010 Bari. ISIE2010 proceedings p.3498-3504 ISBN/ISSN 978-1-4244-6391-6

Wireless Communication Protocol for Agricultural Machines Synchronization and Fleet Management

Alfredo Revenaz*, Massimiliano Ruggeri**, Massimo Martelli**
*University of Ferrara Electronic Engineering Faculty
alfredo.revenaz@unife.it
**IMAMOTER – C.N.R.
Institute of Agricultural and Earthmoving Machines - Italian National Research Council
m.ruggeri@imamoter.cnr.it

Abstract - **In the last years, Precision Farming and Farm Automation in Agricultural field, have been requesting short range wireless communication systems for machine synchronization and long range wireless communication for fleet management, and for production and task control in real time. Existing protocols can meet the requirements only partially: different legislation and product availability in different countries make it difficult to identify a solution for the global market. TCP/IP protocol and GPRS/GSM networks meet the requirements for long range communication, but not for real time communication needed for short range communication. The paper presents a comprehensive proposal based on UDP and TCP/IP protocols with embedded controls, apt to create a quasi-isochronous wireless communication for machine synchronization.**

I. INTRODUCTION

The adoption of electronic systems in Agricultural machines control is related to powertrain management and Implements function management [1]. Downright distributed control systems were designed and standard wired protocols were conceived, in order to allow interconnectivity between Agricultural Tractors and Implements from different manufacturers. ISOBUS protocol allows real time control in machines connected through mechanical, hydraulic, electrical and electronic connections [2]. In the last years more components have been added to the ISOBUS protocol, related to automated functions, automated tasks and autonomous driving from the machine automation standpoint, and related to farm management and working session optimization, from the information management and precision farm standpoint. If at single machine level the system appears as well defined, and the wired CAN communication protocol is a good compromise for distribute machine control, a comprehensive standard must be identified and tested in case of machines synchronization needs. In fact the wide and diversified market of agricultural machines and the high number of different field treatments, need to consider working sessions with more than a machine involved. The field and tasks characteristics impose a wireless communication media and protocol, due to the mission of machines. The structure of the major world companies in agricultural machines production imposes a world standard protocol. Last but not least, the control tasks impose a real time protocol in order to synchronize the machine involved in the job. The basic idea is to customize the Wi-Fi 802.11 compliant standard, whose frequencies are reserved in the whole world, designing an isochronous protocol, able to transfer both real time messages with a dedicated protocol and other kind of information with the standard TC/IP suite.

II. MISSION DESCRIPTION

The machine mission in synchronized operations in the field is described, in order to define and understand the new protocol requirement. As an example, let us consider the automatic loading of a trailer behind a tractor during combine harvesting (Fig. 1). The combine harvester makes contact with the trailer, measures the height of the grain level in the trailer and determines also the position of the spout in relation to the length of the trailer. The combine commands the tractor in front of the trailer to stay synchronized, and it slows down or speeds up. It sends a notification when the trailer is filled and stops unloading.

The combine measures the distance between the sensor on

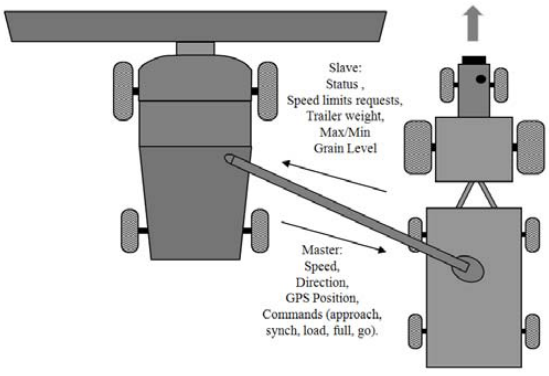


Fig. 1. Tractor/Trailer Load from a Combine Harvester

the spout and the grain level in the truck. Based on information of the minimum and maximum grain level in the trailer, the height of the trailer, sent from the tractor, and the height of the spout acquired directly from sensors, it determines the actual fill level on the position in the trailer.

Thus the typical job in the field requests a master of the task, that communicates parameters and commands to one or more slaves. The slaves' goal is to follow the master at a defined direction and speed, in real time. As a first assumption all the slave machines must be provided with an autonomous driving system or, as a minimum requirement, with a driver interface for an assisted driving. The communication system between machines must provide a real time communication, in order to activate a distributed control task, ensuring that machines are hooked during the programmed operation.

III. REFERENCE PROTOCOLS

The protocol needed by short range communication for machine synchronization purpose has requirements related to real time and safety of communication, due to the need for information reliability and for controlled maximum delay.

Similar problems were analyzed and sometimes solved in other fields of applications, like for example digital audio and video or cars, but it's difficult to find a solution that fits all the requirements of this case study (see [3] to [7]).

From the network topology and management standpoint, Ad Hoc networks seem to be the right solution, in order to manage the machines once they have come in the network area, to start participating to the task, and in order to manage the possible lack of communication, due to electromagnetic disturbances or an excessive distance between a machine and the other network participant. Network management messages must not affect the real time communication between other network participants, that could be already involved in synchronized operations; then a message priority management for both outbound and inbound data flows must be designed, in order to define real time classes as a function of the maximum delay of data to be transferred.

All protocols analyzed in literature, are based on the exploitation of TCP/IP suite functionalities, and many run on Intel based hardware platforms, typically derived from PC, that support complete and customizable Operating Systems, like Linux, but miss the typical real time characteristic of embedded systems.

Even if other protocols, like [8], were afterwards designed for real time data transfer, none of them provides the requested safety functionalities. In fact TCP/IP even if endowed with advanced functionalities (like e.g. reliability, flow control, packets reordering, connections, sessions, etc), was originally designed for applications without strict real time constraints; therefore TCP/IP presents some criticalities concerning traffic management and packet latency.

The idea of using the TCP/IP suite, to take advantage of the high compatibility with other protocols related to fleet management, precision farming and data transfer in general, requires an extra analysis, on the lower levels of the TCP OSI Stack, to assure correct behavior of the system.

The UDP protocol presents some interesting characteristics, but the "best effort" protocol functionality does not guarantee the required reliability and safety integrity level for the typical application, thus requiring some higher level control strategy that will be described. Anyway UDP appears as a transport protocol that fits the real time requirement. The comparison of the two protocols will explain the basic choices made:

1. No connection establishment. TCP uses a three-way handshake before it starts to transfer data. UDP just transmits data without any formal preliminaries. Thus UDP does not introduce any delay to establish a connection.

2. No connection state. TCP maintains connection state in the end systems. This connection state includes receive and send buffers, congestion control parameters, and sequence and acknowledgment number parameters; moreover, these buffers seem not to be user flushable. This state information is needed to implement TCP's reliable data transfer service and to provide congestion control. UDP, on the other hand, does not maintain connection state and does not track any of these parameters. TCP-like connections probably are not fundamental for real time transmission of synchronization data, and it's possible to choose UDP, adding only functionalities really needed, and tailor designed for the application.

3. Small segment header overhead. The TCP segment has 20 bytes of header overhead in every segment, whereas UDP only has 8 bytes of overhead.

4. Unregulated send rate. TCP has a congestion control mechanism that throttles the sender when one or more links between sender and receiver become excessively congested. This throttling can have a severe impact on real-time applications, which can tolerate some packet loss but require a minimum send rate. On the other hand, the speed at which UDP sends data is only constrained by the rate at which the application generates data, the capabilities of the source (CPU, clock rate, etc.) and the bandwidth. We should keep in mind, however, that the receiving host does not necessarily receive all the data, a significant fraction of the UDP-transmitted data could be lost due to buffer overflows, or bandwidth limiting or simply wireless transmission problems. Thus, using UDP means that is necessary to design some new functionalities that can manage packet loss. These functionalities must be probably different from what is already implemented in TCP because the field of applications and the requirements are different.

The information transfer protocol chosen solves the communication problems once the communication session has started and the network and its participants are connected and active; the related network initialization and maintenance can be performed by a dynamic mesh ad hoc network. Many protocols can be found in literature but most of them are subject to latency problems. A short description can help to understand the choice operated ([9] to [12]).

Pro-active (table-driven) routing: this type of protocols

maintains fresh lists of destinations and their routes by periodically distributing routing tables throughout the network. The main drawbacks of such algorithms are: the respective amount of data for maintenance and the slow reaction to restructuring and failures.

Reactive (on-demand) routing: this type of protocols finds a route on demand by flooding the network with Route Request packets. The main drawbacks of such algorithms are the high latency time in route finding and the excessive flooding can lead to network clogging.

Flow-oriented routing: this type of protocols finds a route on demand by following present flows. One option is to unicast consecutively when forwarding data while promoting a new link. The main drawbacks of such algorithms are: it takes a long time when exploring new routes without a prior knowledge and it may refer to entitative existing traffic to compensate for missing knowledge on routes.

Hybrid (both pro-active and reactive) routing: this type of protocols combines the advantages of proactive and of reactive routing. The routing is initially established with some proactively prospected routes and then the demand from additionally activated nodes is served through reactive flooding. The choice for one or the other method requires predetermination for typical cases. The main drawbacks of such algorithms are that advantage depends on amount of nodes activated and that reaction to traffic demand depends on gradient of traffic volume.

Adaptive (situation-aware) routing: is similar to Hybrid routing (with the same drawbacks), but the choice of reaction must be supported by some metrics.

IV. Protocol Structure

From the message standpoint the protocol encapsulates the payload in the UDP message, in order to use the service provided by UDP protocol without inheriting the traffic management provided by TCP/IP, that is not compliant with the application real time constraints. The basic idea is to define a master-slave protocol, where the main machine represents the network master and the reference machine for the tasks to be performed in the field, able to assign IP address to the network participants. The master machine sends at defined timing the requests and commands to the slave machines, and asks for parameters or data related to the task in progress. Sent data and commands are part of a distributed control task and the messages timing must be compliant with maximum delay conditions, that are to be guaranteed by the network structure.

UDP messages seem to fulfill these requirements and the communication structure entrusts to UDP messages, sent both by master and slaves with a defined task timeout, all real time information to be transferred.

The proposed structure implements a communication using UDP packets, where the payload is encapsulated in UDP packets; UDP packets are generated by the protocol, using the standard TCP/IP protocol functions called by the user application, with an higher Priority than all the other messages that can coexist in the system due to different communication session over the network.

The Priority concept is also a standard concept in the TCP/IP stack and nothing is added by the protocol to the standard TCP/IP Stack on this side.

UDP packets are used in short range communication for synchronization purposes, but also in long range communication for fleet management purposes, in order to periodically send a small payload (maximum 576 bytes as explained below) to one or more servers, in order to reduce the protocol overhead with respect to TCP packets, and thus the communication cost, over a GPRS/GSM/UMTS/3G connection to the Internet (using a cellular phone or modem connection). TCP/IP packets can also be used in the long range communication protocol for datalogging and data collecting purposes, for example in order to transfer the files contained in a *file server* (as defined on ISO 11783 standard) to the farm server, or the complete diagnosis *freeze frames* image to the Tractor manufacturer server in case of fault of the machine.

V. Packet Structure

As explained above the UDP protocol does not provide any flow control mechanism and communication reliability is entrusted to a basic payload CRC coupled with the IP header CRC provided by the IP protocol.

Maximum packet length is limited in order to be less than the ‘minimum reassembly buffer size’, this way message fragmentation is completely avoided. Thus the maximum packet length results in:

- 576 bytes for IPV4 protocol version, and
- 1500 bytes per IPV6 ([15], p. 57).

Applications typically use 512 bytes for UDP payload in order to avoid problems or overload if Options are used on IP Packet (safety margin). The packet must be sent with the entire payload at a defined time, without delay due to network management or fragmentation, therefore the buffer size is chosen in accordance with the maximum packet size that is not affected by the fragmentation strategies of the standard UDP protocol. The message payload is specifically designed for each packet, in order to fit exactly the amount of data needed, with no added overhead thank to multicast protocol characteristic. The basic UDP packet structure presented can be customized depending on the packet function to be served. As an example, the presented packet is conceived for a synchronous communication for machines and task synchronization purposes, then, as described in the next paragraph, a Running number is added in order to control packet loss, a Timestamp and a Time Expiration are added in order to control synchronization of packet reception and Timeouts, and a CRC is added in order to increase the robustness of packet data integrity control. This communication represents the maximum safety level requested by the applications: Isochronous communication with Safety requirements; all the other types of communication can be derived from this model.

In case of need for Fleet Management, the packets will probably result in small packets with no safety requirements, less timing constraints and a considerable importance of data integrity, but with a high importance of communication cost, because of the continuous need for data transmission (for example a packet @ 5 or 10 seconds of transmission period); this kind of packet can be managed using UDP, that offers a lite CRC in the packet with a minimum protocol overhead, and that allows a single sender and multiple receivers with a unique socket opened on the net, in order to allow the Fleet Management for different machines or Functions from a unique Gateway and a unique active physical connection to more servers through the Internet. This way the communication cost is minimized, data integrity is quite assured by the UDP CRC and the data flow can be controlled using a running number. Another type of communication is represented by the transfer of files collected in the storage memory of the data logging units embedded on the machines. These data can result in big amounts of Kbytes that the logging unit is programmed to transfer to a server. These data could be transferred to the server via a Wi-Fi connection once the Machine has come back to the farm, with no communication cost and without risks related to the coexistence with short range communication for synchronization purposes, or, in the worst case, these data could be transferred to the server using a GSM/GPRS/UMTS/3G communication via a modem through the Internet, when the machine is on field and with some risk of coexistence with short range communication for synchronization purposes. In these cases any standard protocol over TCP/IP can be used (FTP or SFTP through a client and a socket on the server, or with a client/server connection to a database in the farm or in the manufacturer site).

This scenario will be used in the tests in order to consider the worst condition for real time communication.

VI. Protocol Services For Traffic Management

The way to create the new set of protocols apt to satisfy the current requests for the agricultural machines short and long range communications, are basically two diametrically opposite to each other:

1. The first one leads to a very basic protocol that includes only the basic communication functions; in this protocol all the functionalities and the services related to Priority management, timestamps and timeouts management and control, CRCs, messages time scheduling, bandwidth and network traffic control, are entrusted to the application, and then every manufacturer can implement all these functions as it prefers, with a resulting very simple protocol and very small documentation.
2. The second one defines a protocol that regulates and standardizes all the communication functions and all the functions responsible for any service; the resulting protocol is a more complex protocol that ensures a better compatibility between machines with a likely better regulated traffic, safer, but with some throughput constraints, due to the implemented controls that are currently generic and not yet tested on real applications.

The first solution leaves designers free to define task timing and allocate bandwidth for the different protocols, with the advantage of tailoring the communication timing on the application needs, but with the uncertainty of the performance of the other different machines in the task. The second one, on the contrary, regulates all traffic and then guarantees the compatibility and the performances of all machines in the task, if compliant with the protocol.

Any intermediate solution is possible and some task force for protocol mission definition, implementation and tests is necessary, in order to define the protocol performance boundaries. In the test cases the first solution was adopted in order to test the worst condition for real time traffic, that results affected by the presence of many independent UDP real time flows and FTP traffic without any strategy for FTP bandwidth restriction.

VII. Safety Management

The packet encapsulated in the UDP payload is generated taking into account the typical payload in agricultural machines tasks and some basic principles related to the safety of network communication as stated by the normative directives [13] and [14].

In order to increase the protocol reliability and set up a basic flow control, the rules described in [14] are implemented and protocol safety and reliability are guaranteed by the adoption of the following set of control mechanisms:

1. A *Running Number*,
2. A *Timeout* for all messages, coupled with a *Time Stamp* transferred in the messages,
3. A *Software CRC* (32 bit CRC) added to the one provided by the standard protocol,
4. An *Explicit Acknowledgement* for all Safety Related messages.

These measures safeguard against the most common communication errors, as explained in Table I, in order to provide a more reliable protocol. The safety level shall be defined as a function of the message and of the criticality of the contained information for the task of the agricultural machines. The payload structure refers to the CAN messages characteristics, in order to transfer the data from the in-machine network to the wireless network task with a minimum computational load for the CPU and with all control structures like for example an 8-level priority embedded in the message (Table II).

VIII. Test Description

At the current stage all tests were performed on both PC and embedded platforms, all Intel based, with standard Wi-Fi (802.11g) hardware and Linux Operating System, Kernel with or without Real Time patch; all processes, created by

Table I
Communication errors and countermeasures

Transmission error	Measures per message							
	Running number	Time stamp	Time expiration	Reception Acknowledge	sender/receiver Identification	Data integrity assurance	Redundancy with cross check	Different data integrity assurance systems
Repetition	x	x					x	
Loss	x			x			x	
Insertion	x			x[a]	x[b]		x	
Incorrect Sequence	x	x					x	
Message Falsification				x		x	x[d]	
Retardation		x	x[c]					
Coupling of SR and non-SR information				x[a]	x			x

software written in C language, were running in user mode (not kernel mode).

The basic tests were performed to show that it's possible to send a ten thousands UDP packets, each one carrying 500 bytes of payload, with a 10 ms period (100 Hz repetition rate), with an average delay of a few milliseconds, in presence of contemporaneous traffic at lower priority simulated by an FTP file transfer over TCP that tried to saturate the bandwidth.

The machines under test are two hosts with data send and receive capability and a protocol analyzer, able of reading and interpreting TCP/IP and lower/higher layer protocols traffic, connected to a wireless 802.11g network in Ad-Hoc mode (see Fig. 2). The 'Micro' host is an embedded micro-PC from Lex Computech with a VIA C7 nano 1 GHz CPU, 1GB DDR2 RAM, an 80GB hard disk and a CentOS 5.3 Linux O.S. The Wi-Fi hardware is an Intel BG2945 wireless mini-PCI card with two +5 DB antennas in diversity mode.

The 'Macro' host is a desktop PC with a Pentium 3 933MHz CPU, 512MB SDRAM, an 80GB hard disk and the same CentOS 5.3 Linux O.S. The Wi-Fi hardware is an Intel BG2200 wireless mini-PCI card with a PCI adapter board and two +5 DB antenna in diversity mode.

Table II
Encapsulated Message Structure

	Priority	Machine Identifier	Function Identifier	Sequence Number	Time Stamp	Flags	Data (Payload)	SW CRC
Length (Bytes)	1	4	4	2	4	1	576 – other fields – UDP Header – IP Options (if any)	5

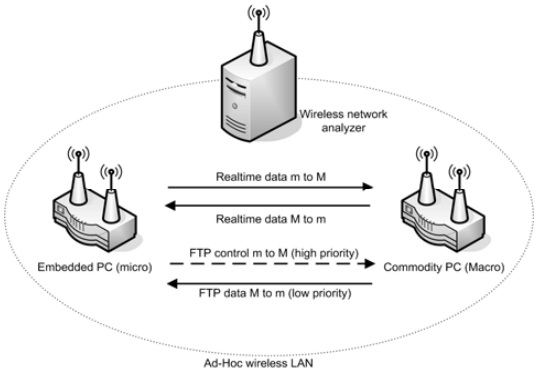


Fig. 2. Test site schema

The hosts operating systems were intentionally not optimized for the application, except for the use in some tests of an RT kernel, left in default configuration, to implement a worst case scenario mainly in terms of latency. Some tests were made assigning higher priority to sending and receiving tasks, to evaluate the effects of different task scheduling priorities at the hosts on transmission timings.

The traffic was generated via Sockets Networking API on Linux custom made client and server software, running in user mode and written in C to 'emulate' an isochronous real-time protocol.

Standard FTP client and server (VSFTPD –Very Secure FTP Daemon) were used to test data transmission in presence of a large amount of TCP normal- priority traffic. FTP traffic was generated by transferring a file from Macro to Micro host, so that it was only limited by link capacity. Priority of FTP client and server tasks were not modified, but left to default. UDP packets had a size of 508 bytes, with a 500-byte payload containing some data simulating a real-time isochronous protocol (i.e. Message ID, Function ID, Priority, Sequence number, Sender Timestamp, and a generic payload to complete the 500-byte block). The custom made application transmitted 10.000 UDP packets with a period of about 10 ms for every run, with or without higher task priority on both sender and receiver, with or without IP low-delay Type of Service (TOS) enabled on UDP traffic, with or without contemporaneous FTP traffic.

Task priority was controlled setting *nice* level for sending and receiving tasks at system level, using a nice value of -20 (corresponding to a priority of 0, whereas normal task priority is 20 on CentOS O.S). Linux kernel implementation honors TOS bits only on outbound traffic, so the benefits of traffic priority are limited; the performance could be improved activating priority also on inbound traffic, which requires a Linux kernel patch, currently under development.

Some test results are presented in the following figures.

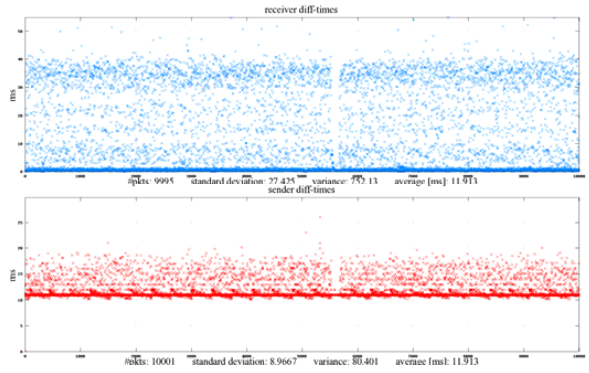


Fig. 3. Worst case (standard TCP)

All diagrams show the time gap (in milliseconds) between consecutive sent/received packets, reporting also average value, standard deviation, variance, and number of received packets, which can differ from the sent number due to the wireless link. In some cases graphs axes are rescaled to facilitate reading. In all test runs described hereinafter distance between hosts was 10 m, link was signaled at theoretical 54 Mbit (802.11g) and signal quality was 90% or better.

Additional tests were carried out, with different hosts and hardware, under different site conditions. In particular open-air tests with distances of about 100 m gave very similar results.

The results of only a small part of the performed tests are shown here.

Fig. 3 refers to the worst case, resulting from the transfer of a single data flow with a 10 ms period, no traffic priority, default task priority, non RT kernel and contemporaneous FTP traffic, as described before.

The time gaps have a big variance, for both send and receive tasks, with strong effects of the superimposed FTP traffic at network level and host level.

Fig. 4, 5 and 6 instead show the effects of the implemented traffic priority strategies, in case of multiple traffic flows, in the particular test case 2 flows from Micro to Macro and 2 in the opposite direction and with or without concurrent FTP traffic. In these figures a detail of 1000 messages in tests of 10.000 messages are presented, in order to clarify the timestamps and the maximum deviation from the expected time for the networked communication and control task.

All the previously listed strategies, designed to maximize performance, were active in the considered test runs.

Further tests, carried out with up to 10 concurrent flows showed that system performance tends to remain similar. FTP traffic is directed from Macro to Micro host in all tests and, due to the default settings of applications that implement the FTP service, has a high priority in the control channel and a low priority on the data channel, the latter being quantitatively predominant.

Fig. 4 shows the ideal case, with no concurrent traffic and in fact the receive time gaps in both directions are very close to the average value, with minimal data dispersion and very low standard deviation. Fig. 5 and 6 show the results in case of contemporaneous FTP flow, as described before.

In particular Fig. 5 highlights the flows received in both directions, the first 2 from host Micro, the last 2 from host Macro. These graphs show that data dispersion is extremely reduced with respect to the worst case.

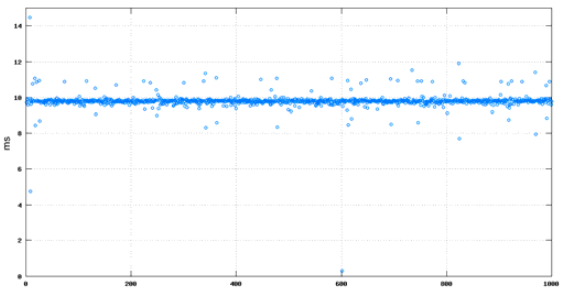
In the 2 upper diagrams of Fig. 4 a 'splitting' occurs in the plotted data, probably due to the lack of priority control for inbound traffic, that causes the queuing of long trains of FTP packets between 2 consequent high priority traffic packets, and thus a drop in the graph. A Linux kernel patch which will solve this problem is in its final stage of development.

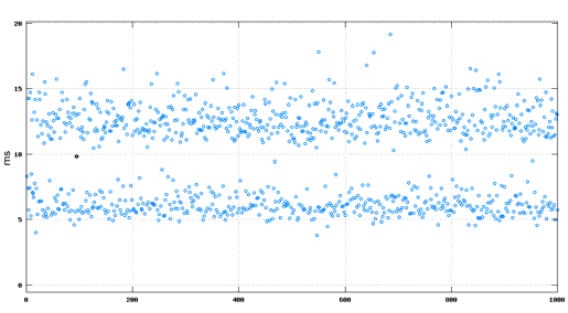
Fig. 6 shows the results for send times, which already display an acceptable performance. It can be said that from the transmission standpoint most problems are already solved, even in presence of concurrent FTP traffic.

IX. Result Discussion & Conclusions

Despite the complexity of customization of a Linux system and the lack of Inbound messages priority management, the test results suggest that the protocol designed and here proposed can be eligible for a good solution for the short range communication in agricultural applications for machine synchronization purposes. The low message loss rate, around 0,03%, and low message delay even in case of 10 contemporaneous real time data flows and a concurrent FTP without bandwidth limitation, fit the application needs; the message delay and receivers overload should be reduced and minimized using the Inbound traffic priority management, that is currently under development.

Similarly the mesh network management traffic should not affect the already established real time traffic because it should be scheduled at lower priority.





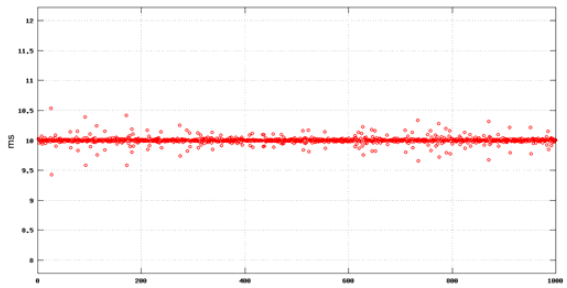


Fig. 6. Sender real time traffic with FTP

ACKNOWLEDGMENT

A special thanks to Prof. Daan Goense, Convener of the ISO TC023/SC19/WG5 Working Group, for supporting and encouraging us in the development of this new standard and for sharing all his experiences in this complex field of applications.

REFERENCES

[1] SAE J 1939 (all parts), SAE International, U.S.A., 2006.
[2] ISO 11783 (all parts) *Tractors and machinery for agriculture and forestry — Serial control and communications data network*, 2009, ISO Geneva, Switzerland.
[3] C. Mbarushimana, A. Shahrabi, *Comparative Study of Reactive and Proactive Routing Protocols Performance in Mobile Ad Hoc Networks*, 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07), 2007.
[4] Fu Yongsheng , Wang Xinyu , Li Shanping, *Performance Comparison and Analysis of Routing Strategies in Mobile Ad Hoc Networks*, 2008 International Conference on Computer Science and Software Engineering, December 2008.
[5] Glaucia Campos, Gledson Elias, *Performance Issues of Ad Hoc Routing Protocols in a Network Scenario used for Videophone Applications*, Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 9, 2005.
[6] Samir R. Das, Robert Castaneda, Jiangtao Yan, Rimli Sengupta, *Comparative Performance Evaluation of Routing Protocols for Mobile, Ad hoc*, Seventh International Conference on Computer Communications and Networks (ICCCN '98), October 1998.
[7] Toby Xu , Ying Cai, *Streaming in MANET: Proactive Link Protection and Receiver-Oriented Adaptation*, 2007 IEEE International Performance, Computing, and Communications Conference, April 2007.
[8] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications*, The Internet Society, July 2003.
[9] C. Mbarushimana, A. Shahrabi, "Comparative Study of Reactive and Proactive Routing Protocols Performance in Mobile Ad Hoc Networks", 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07), 2007
[10] Fu Yongsheng , Wang Xinyu , Li Shanping, "Performance Comparison and Analysis of Routing Strategies in Mobile Ad Hoc Networks", 2008 International Conference on Computer Science and Software Engineering, December 2008.
[11] Glaucia Campos , Gledson Elias, "Performance Issues of Ad Hoc Routing Protocols in a Network Scenario used for Videophone Applications", Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 9 , 2005.
[12] Toby Xu , Ying Cai, *Streaming in MANET: Proactive Link Protection and Receiver-Oriented Adaptation*, 2007 IEEE International Performance, Computing, and Communications Conference , April 2007.
[13] *IEC 61508, Functional safety of electrical/electronic/programmable electronic safety-related systems – Parts 1-7*, IEC Organization, Geneva, Switzerland, 2007.
[14] *ISO 15998, Earth-moving machinery — Machine-control systems (MCS) using electronic components — Performance criteria and tests for functional safety –*, ISO Organization, Geneva, Switzerland, 2009.
[15] W. Richard Stevens, Bill Fenner, and Andrew M. Rudoff, *Unix Network Programming, Volume 1: The Sockets Networking API* (3rd Edition) Addison-Wesley Professional Computing Series - Nov 24, 2003.

2. International Symposium on Industrial Electronics (ISIE) 2011. Sottomesso il 31-12-2010. Accettato il 23-02-2011. Sarà presentato al congresso dal 27 al 30 giugno 2011.

Low Latency WI-FI Real-Time Protocol for Agricultural Machines Synchronization Using Linux RT Kernel

Alfredo Revenaz*, Massimiliano Ruggeri**, Velio Tralli*
*University of Ferrara Electronic Engineering Faculty
alfredo.revenaz@unife.it, velio.tralli@unife.it
**IMAMOTER – C.N.R.
Institute of Agricultural and Earthmoving Machines - National Research Council of Italy
m.ruggeri@imamoter.cnr.it

***Abstract* – The production needs and the more complex working sessions in agricultural field ask for machine cluster in order to achieve a parallel and complete treatment of crops and fields. All heterogeneous or homogeneous machines of the cluster, both in series and in parallel, have to be synchronized to match safety requirements and to comply with task control requirements. The synchronization specifications require a real time wireless communication protocol affordable, safe and with low latency, in order to ensure a minimum guaranteed throughput apt to synchronize the machine work and travel. The paper describes the characteristics of a quasi-isochronous Wireless transmission protocol based on IP/UDP under Linux Operating System and the protocol evolution with respect to what reported in previous work[1]. In this paper the performance, from the throughput and the statistic point of view, is explained in a quantitatively, based on tests performed using a complex and fully automated test rig. A solution based on task and messages priority management is proposed, using new Linux Real Time Kernel standard features explained in [2] and [3]. The obtained test results suggest that the protocol architecture proposed is a promising solution for low latency communication in both industrial and mobile applications.**

I. INTRODUCTION

The main idea described in [1] helped agricultural machines constructors to become aware of needs and performance in machine synchronization communication protocol design. The protocol development and the performance tests described here are related to the renewed interest and specification discussed in ISO international meetings with the largest agricultural machines manufacturers worldwide.

As explained in [1], a comprehensive standard must be identified and tested in case of machines synchronization needs. In fact the wide and diversified market of agricultural machines and the high number of different field treatments, need to consider working sessions with more than a single machine involved.

The global structure of the major companies worldwide in agricultural machines production forces a worldwide standard protocol. Last but not least, the control tasks impose a real time protocol in order to synchronize the machines involved in the job. The basic idea is to use the Wi-Fi 802.11 compliant standard, whose frequencies are reserved worldwide, designing an isochronous protocol, able to transfer real time messages with a dedicated protocol and to transmit other kind of information with the standard TC/IP suite. The first implementation of that protocol gave acceptable results in terms of real time, in case of absence of other TCP traffic superimposed, and in terms of acceptable number of packet loss, but improvable in terms of real time, especially due to the fact that the software strategy was not optimized in function of packet management.

Moreover t was not possible to rate accurately some performance due to the different hardware platforms computing power and architecture. A new revision of the protocol and data management, and a new version of Linux RT Kernel (2.6.33.7 Revision, whilst the previous was 2.6.24.7) configured with all Real Time features activated, demonstrates increased performance in terms of packet loss and real time of packets delivery, both with and without several overlapped data in the same wireless channel.

II. APPLICATION SCENARIO

In order to define and understand the new protocol requirement ,the machine mission in synchronized operations in the field need to be described. As an example, the parallel

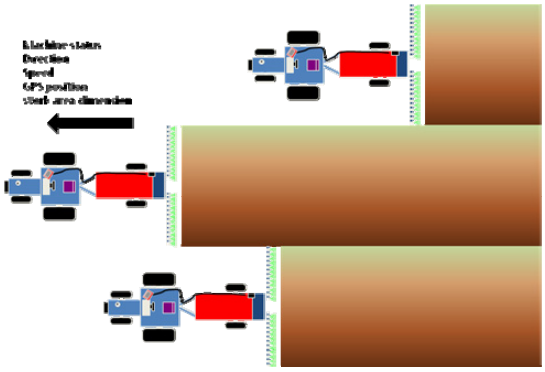


Fig. 1. Cluster of homogeneous machines in a synchronized work session

work session of some agricultural machine during a spraying field treatment is set-up by different phases (Fig. 1):

- The cluster of homogeneous machines starts a communication session in a ad hoc WI-FI network. From the protocol management point of view the role of Master of the work session is given to a machine from the operator.
- The master machine starts a communication session where the Machine status and other data related to task to be performed are shared with other machines of the cluster.
- The data are related with GPS data of the master machine and with field treatment data. Other data are related with the machine configuration in order to allow the other machines to adjust the position during the working task, avoiding overlapping of the treated field area or loss of field treated areas. One of the data is related to the implement dimension that define the machines relative position in the field.

The dynamic of data transmission has to consider the dynamics of the machine direction change and speed, or more precisely the implement direction change, that can be evaluated in hundreds of milliseconds.

Due to this specifications, a minimum set of information must be transferred from the main machine to the nearby machines and, similarly the same situation must be created also for the other machines in a communication that can be configured initially in broadcast, in order to create the machine configuration, based on GPS machine position, and, in a second phase, with a point to point communication without any need to hopping the information.

Once the network is configured, the task can be started and the communication must ensure a continuous stream of information apt to create a really networked control task, ensuring that machines are hooked during the programmed operation.

The dynamics of the system were evaluated and a throughput of 500 payload byte at 50 / 100 ms for each machine status and commands were defined as a minimum requirement for machine synchronization purposes.

Before defining the data contained in the messages, a low level WI-FI protocol must be identified, designed and tested, in order to assure a maximum delay in data transfer, compatible with the system and control requirements.

Other task types are defined and all offer the same real time needs to the communication specification. Due to multiple functionality implemented in a typical control unit installed in agricultural machines, some traffic type can be defined, each one with a typical real-time and safety level requested. An agricultural machine, at top of the range, can be equipped with a fleet management system that allow the continuous link with the farm for working session control and continuous working session program re-plan and logging; the current typical link for fleet is a GPRS/UMTS connection through the internet. The working program, the fleet management data and so on must be considered non real-time data, nevertheless the traffic generated cannot be neglected, because it is a TCP traffic that will be managed by the same system resource.

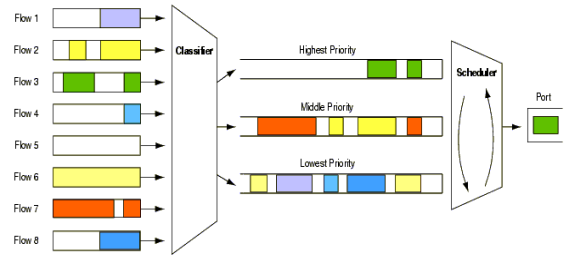


Fig. 2. Outbound queue Priority management

III. THE PROTOCOL

The protocol needed by short range communication for machine synchronization purpose has requirements related to real time and safety of communication, due to the need for information reliability and for controlled maximum delay.

Even if other protocols were designed for real time data transfer, none of them provides the requested functionalities.

The idea of using the TCP/IP suite, to take advantage of the high compatibility, required an extra analysis on the lower levels of the TCP OSI Stack, to assure correct behavior of the system.

The UDP protocol presents some interesting characteristics, and after some tests it appears as a transport protocol that matches the requirement of a quasi-isochronous real-time protocol, provided that some strategies are adopted.

Using UDP means that it is necessary to design some new functionalities that can manage additional requirements in terms of real-time features. These functionalities must be different from what is already implemented in TCP, because the field of applications are different. From the tests done until now all requirements cold be fulfilled encapsulating a new packet structure in the standard UDP, e.g. using UDP merely as a transport under some appropriate conditions.

This choice permits to use the service provided by UDP protocol without inheriting the traffic management provided by TCP/IP, that is not compliant with the application real time constraints.

UDP packets are generated by the protocol, using the

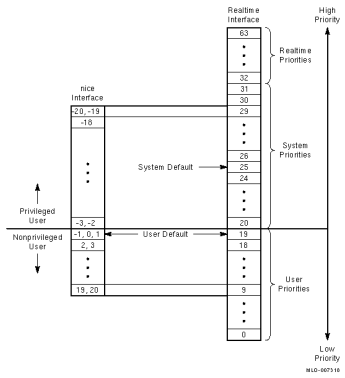


Fig. 3. Priority management with Nice and Realtime kernel interface

standard TCP/IP functions called by the user application, with a priority higher than all the other messages that can coexist in the system due to different communication session over the network.

The Priority concept is also a standard concept in the TCP/IP stack and nothing was added by the protocol to the standard TCP/IP Stack on this side until now (Fig. 2). In the future, a smart system of priority management could be imagined and implemented to guarantee separation between transmission of high real-time traffic and lower priority user traffic.

Focusing now on the short range communication for synchronization purposes, some data are reported for packet structure; the maximum packet length is limited in order to be less than the 'minimum reassembly buffer size'. This way message fragmentation is completely avoided. Thus the maximum packet length results in:

- 576 bytes for IPV4 protocol version, and
- 1500 bytes per IPV6 ([15], p. 57).

Applications typically use 512 bytes for UDP payload in order to avoid problems or overload if Options are used on IP Packet (safety margin).

In order to increase the protocol reliability and set up a basic flow control, some other data are added to the standard general purpose UDP payload:

1. A *Running Number*,
2. A *Timeout* for all messages, coupled with a *Time Stamp* transferred in the messages,
3. A *Software CRC* (32 bit CRC) added to the one provided by the standard protocol,
4. An *Explicit Acknowledgement* for all Safety Related messages.

IV. Test Site And System Configuration

The test bed is modified with respect to what described in [1] and more machines were added, all with the same characteristics, in order to test the physical limit of the protocol, without any limitation due to hardware computing power or architecture. Afterwards some PC based platform were added in an indoor test site, where machines were connected through a wired Ethernet, in order to synchronize and plan tests, and moreover wireless communication was provided by a hardware PCI Wireless N TP-Link TL-WN851N, compatible with the 802.11 b/g/n standard, and that is provided by two antennas, with MIMO technology and Atheros chipset.

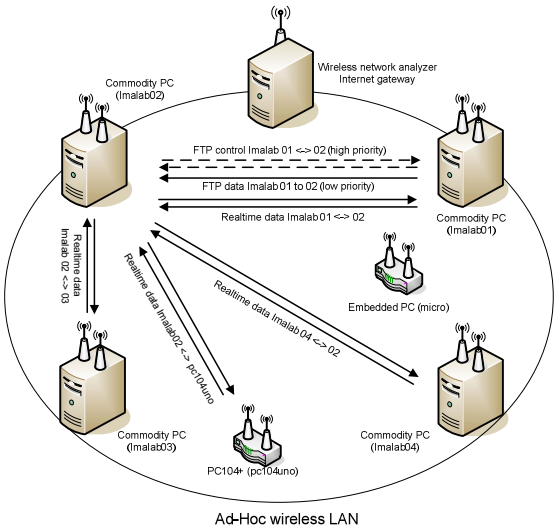


Fig. 4. Test site schema

The hardware was chosen based on Linux Kernel support and compatibility. The hardware supports both the infrastructure and the ad hoc network, while during the system set up it was found that support for the *n* technology in Atheros drivers in ad hoc mode is still immature.

The PC based platforms are all configured using a CentOs Linux distribution with a kernel version 2.6.33.7 with Real Time preempt Patch that has been recompiled disabling the Advanced Power Management (APM) and Advanced Configuration and Power Interface (ACPI) in order to avoid uncontrolled CPU load during test sessions. The standard Linux kernel only meets soft real-time requirements: it provides basic POSIX operations for user space time handling but has no guarantees for hard timing deadlines. With Real-time Preemption patch and a new clock event layer with high resolution support, the kernel gains hard real-time capabilities. The RT-Preempt patch has raised quite some interest in industry. Its clean design and consequent aim towards mainline kernel integration makes it an interesting option for hard and firm real-time applications, for example in the industrial control sector. The RT-Preempt patch converts Linux into a fully preemptable kernel. This result is achieved with:

1. Making in-kernel locking-primitives preemptable though reimplementation.

2. Critical protected sections now preemptable. The creation of non-preemptable sections (in kernel) is still possible with a dedicated primitive.

3. Implementing priority inheritance for in-kernel spinlocks and semaphores.

4. Converting interrupt handlers into preemptable kernel threads.

5. Converting the old Linux timer API into separate infrastructures for high resolution kernel timers plus one for timeouts, leading to user space POSIX timers with high resolution.

Moreover also the Standard CentOS Linux distribution firewall an SeLinux configuration were disabled as well as all unessential services.

As shown in Figure 4, the test site Was set up using an indoor environment equipped with 6 machines, which were:

1. Four commodity hardware PC, with Intel Pentium 4 3 GHz CPU an 512 MB RAM,

2. One embedded micro-PC with a VIA C7 nano 1 GHz CPU, 1GB DDR2 RAM, an 80GB hard disk. The Wi-Fi hardware is an Intel BG2945 wireless mini-PCI card with two

Table I
Old Protocol Version Receivers Statistical Analysis (mean Values)

Pkt n°	Stream n°	FTP	Standard Deviation	Variance	Pkt loss ‰
10000	2	off	0,1852	0,0343	0,0000
10000	2	on	2,8919	8,3941	55,600
10000	5	off	0,3469	0,1214	0,0200
10000	5	on	2,4959	6,2428	74,450

+5 DB antennas in diversity mode and

3. A PC104+ equipped with a VIA EDEN V4 and a VIA CX700M chipset, 1 GB DDR2 RAM, a 8 GB Flash disk. The Wi-Fi hardware is an Intel BG2945 wireless mini-PCI card with two +5 DB antennas in diversity mode.

4. a network analyzer equipped with WireShark software under Linux O.S.

All units are connected through an ad hoc mode wireless network and through a wired Ethernet connection in order to control the test run and to acquire the test results.

V. Software Description

The software suite was designed using different languages and features depending on the tasks to be performed. The basic application, intended to manage the real time wireless communication, were written in Posix C language, and compiled using GCC as a multi-thread application running in user mode under Linux O.S.

The application is divided into two main parts: the sender process and receiver process.

The Sender process is configured to run at nice level -5 and real-time priority 89, because that value is the maximum priority value suggested for user processes, while priority from 90 to 99 are normally used for system processes, which is recommended not to interfere with. Specific real-time features activated are: the RT priority and process memory lock in order to avoid system swap on hard disk.

A similar configuration was implemented for the receiver process. The receiver is a multithreaded application, composed by two threads running at different priority levels in order to assure the real time communication management at messages reception level. The higher priority thread runs at nice level -5 and real-time priority 89, exactly as the sender process; this process in the initial and final communication session phase is responsible for start and stop packet recognition and for communication session initialization from the software management point of view, and for the lower priority thread activation. During the runtime communication phase it manages the packet reception from the socket and save data on a software buffer.

The lower priority receiver thread, configured at default system priority level for user applications, once initialized opens some files and save data from buffer to hard disk drive; at the end of communication session it closes the files and end the session.

Some configurable batch scripts were designed in order to manage automatically the test session in all machines from the analyzer machine, using wired Ethernet network.

Other batch scripts are used during the test results and post process data analysis phase in order to reorganize the test data and to invoke the Octave software with a specific script that analyzes data from the statistical point of view and generates the test graphics (Figures 5 to 9) and composes the test description sheets in txt in order to collect all configuration data and other useful data for the entire test bed for test characterization. All statistical data are stored in files in *csv* format.

Table II
RT Performance Receivers Statistical Analysis (mean Values)

Pkt n°	Stream n°	FTP	Standard Deviation	Variance	pkt loss ‰
10000	12	off	0,6851	0,4733	0,0000
10000	12	on	2,7445	7,6763	1,4250
10000	30	off	1,5075	2,3155	0,0000
10000	30	on	3,1858	10,1907	3,4800

VI. Test Description

The test plan was designed taking into account all different traffic types and conditions that could occur in a real application, i.e. the real time traffic for machine synchronization and a non real time traffic with potential huge band occupation, like an FTP transmission/reception or a TCP communication for fleet management or data logging through the internet.

The protocol was implemented and the performance analyzed in a real indoor environment.

Standard FTP client and server (VSFTPD – Very Secure FTP Daemon) were used to test data transmission in presence of a large amount of TCP normal priority traffic. FTP traffic was generated by transferring a file from a sender host to the receiver host, so that it was only limited by wireless link capacity. Priority of FTP client and server applications were not modified, but left to operating system default.

The custom made application transmitted 10.000 UDP packets with a period of about 10 ms for every run, with higher task priority on both sender and receiver applications, with IP low-delay Type of Service (TOS) enabled on UDP traffic 'real-time', with or without symultaneous FTP traffic. In all test runs described hereinafter distance between hosts was 20 m, link was signaled at theoretical 54 Mbit (802.11g) and signal quality was 80% or better.

In order to evaluate the protocol performance and the

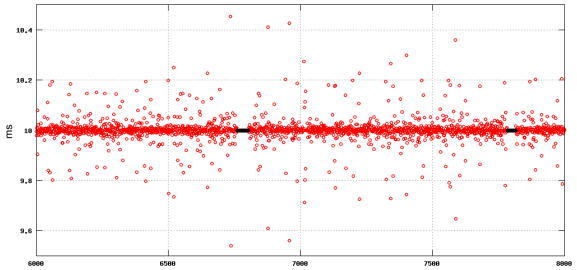


Fig. 5. Old Results Sender with Full Bandwidth FTP Traffic

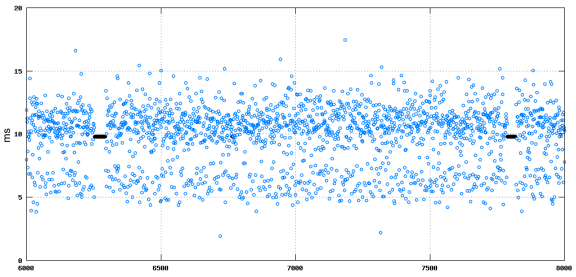


Fig. 6. Old Results Receiver with Full Bandwidth FTP Traffic

improvement with respect to the first protocol implementation in terms of real time, packet loss and maximum throughput available for real time transmission, some parameter were identified and, consequently, the test suite structure was adapted to store all relevant data.

Real time relevant data are basically the time difference between two subsequent data packet sent and received.. To this purpose, both sender and receiver timestamp are logged; the former is considered at packet generation time in the sender machine, in order to include the operative system real time effect in the performance evaluation, and is included in the packet payload, while the latter is read and stored at reception time at application level running as a user process. The machine time synchronization is based on the internet NTP protocol with the same timeserver for all machines; anyway, even if it cannot be assured that the time is strictly identical in all machines, the relevant data for the tests are the time difference between following data from the same source and received from the same destination and then the differences between homogeneous data are not affected by clock difference or absolute time errors.

The relevant data used to calculate performance are:

- Number of machines in the cluster, that define the collision probability due to arbitration in the communication channel;
- Number of data stream between machines in the cluster, that define the traffic level and the collision probability;
- Number of packet loss, that define the quality of transmission, and the effect of the traffic in the receiver units.
- Time difference between two packets sent by sender unit, defined as:

$$\Delta T_{Sj}(i) = T_{Sj}(i) - T_{Sj}(i-1)$$

where S is the Sender, j is the stream index and i is the packet running number.

- Time difference between two packets received by receiver unit, defined as:

$$\Delta T_{Rj}(i) = T_{Rj}(i) - T_{Rj}(i-1)$$

with identical symbol significance except for R that indicates that is the received stream.

These differences are used to calculate the statistical data for performance evaluation.

- The standard deviation:

$$std(x) = \sigma(x) = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1}}$$

Where x is the stream data vector that contains all ΔT_{Sj} and ΔT_{Rj} data.

- The variance:

$$\mathrm{var}(x) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N}$$

- And, using the packet loss number, a packet loss per thousand is calculated.

All statistical data were calculated applying Octave Software directly to the raw data stored from the machines during the tests.

The test were run in two basic configurations: with and without a superimposed non-real time high bandwidth traffic. The worst case for the real time traffic performance is the simultaneous presence of a FTP data transfer without bandwidth constraints.

The positive effect of the Linux RT Kernel in real time task scheduling and execution can be observed in Figure 5 and 7, where a comparison between standard Linux kernel and RT Kernel demonstrates that all packets are processed without packet loss and the Standard Deviation of ΔT_{SJ} is typically two orders of magnitude smaller. This can be noted in the figures vertical axis scale, in ms, that shows packet $\Delta T_{SJ.}$. In figure 5, the most of the packets ΔT_S are between 9,8 and 10,2 ms, but values can be found till to 14 ms, while in Figure 7, referred to the RT Kernel performance, all packets are between 9,50 and 9,52 ms.

Moreover, as explained in Paragraph III and in Figure 2, the Output Traffic Priority is always use,d and the traffic queue allow the real time traffic to be processed always

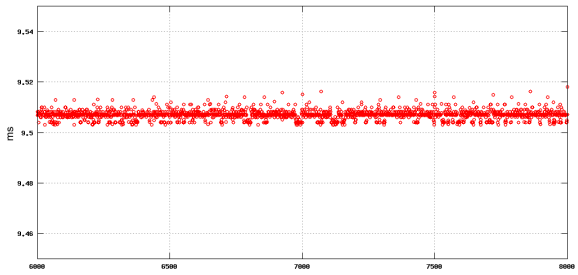


Fig. 7. Sender traffic with RT Kernel and Full Bandwith FTP Traffic

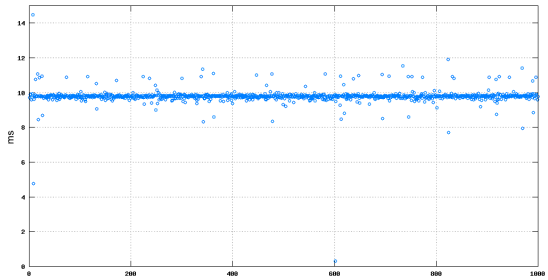


Fig. 8. Receiver, real time traffic without FTP

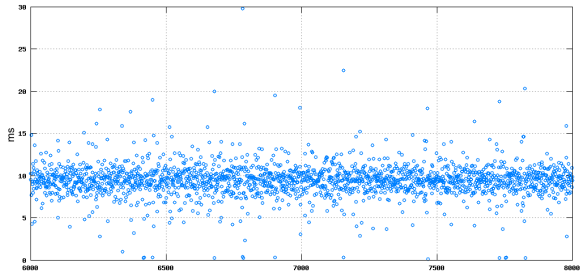


Fig. 9. Receiver traffic with RT Kernel an Full Bandwidth FTP Traffic

before the lower priority traffic, as for example a superimposed FTP communication.

In this condition, where sender schedules and generates packets in a quasi-deterministic way, the transmission channel effect, the receiver communication buffer, the receiver machine task scheduler effect and other minor effects can be evaluated only by the ΔT_{RJ} analysis.

From the pure O.S. point of view, the Figure 8 shows that without any disturbance the communication channel affect in a minimal way the real time performance. In the figure, in fact, can be observed that the ΔT_{RJ} are mostly distributed over the 10 ms line with a typical Standard Deviation of 0,71 ms.

A visible negative effect over real time can be observed in Figure 9, where a traffic detailed view presents a received packets ΔT_{RJ} values dispersion greater than that in Figure 8. Anyway, even if at a first sight it can result similar to the behavior of the non-real time system shown in Figure 6, a closer analysis demonstrates that the performance are 6 times better. In fact, an analysis of performance in Table I and II demonstrates that the RT Kernel system offers similar Standard deviation in a test bed with 30 contemporaneous data stream and 5 concurrent machines in the cluster, while the non-real time system was tested with only two machines in the cluster and with 5 contemporaneous data stream.

The machine number is a significant condition, because of the concurrent use of the wireless radio channel at first. Collision of packets transmission can affect real time performance. Moreover, when two machines only are involved in communication, Type Of Service (TOS) priority managed on output determines the effective order also of received packets; in that situation high priority real time packets are typically served first, thus the lack of inbound traffic priority management of the Linux Kernel is not influent. When the order of inbound packets is determined by the competition of data streams received from different machines, with low and high priority (TOS) packets, the lack of Inbound priority management probably has a relevant effect. For example packet loss in presence of concurrent FTP traffic, as seen in Table II, derives probably from lack of inbound traffic priority management. The real-time kernel was found relevant also to minimize asynchronous events effects. In fact Figure 6, referring non real time kernel, shows two small horizontal marks between index 6000 and 6500, and 7500 and 8000 indicating continuous packet loss. This kind of event disappeared using a properly configured RT Kernel, as explained in [2] and [3], and adapting accordingly sender and receiver applications.

VII. CONCLUSIONS

The paper demonstrates that the problem of real-time isochronous protocol design seems to be solved at least from the packet sending performance point of view, relating to the actual reference application requirements. Performance with 5 machines and 30 data streams in test bed, shows that a system similar to Fig. 4 can fulfill requirements in the real world applications. From the receiving point of view, performance reached by the protocol are good, and considerably better than those presented in [1]. Nevertheless improvements are feasible and they may come mainly from two factors: the first is Inbound Traffic Priority Management - not implemented in the current release of the Linux Kernel - , the second is the 802.11n standard. In this paper all tests were executed using the older 802.11g standard because support for the newer "n" standard is immature in the current drivers in ad-hoc mode. The Inbound Traffic Priority Management is under development, following two separate approaches: a dedicated Linux kernel patch that is in preliminary test phase, and using Linux Intermediate Queuing Device [4], an already developed patch that allows to obtain inbound traffic priority management using a virtual device. IMQ is currently under development and from our side is under test. 802.11n ad-hoc mode support is also under development, and it will be ready in a short time, bringing probably a performance improvement of a factor 10 concerning the wireless transmission channel.

REFERENCES

[1] Alfredo Revenaz, Massimiliano Ruggeri, Massimo Martelli, *Wireless Communication Protocol for Agricultural Machines Synchronization and Fleet Management*, 2010, International Symposium on Industrial Electronics (ISIE 2010), July 2010.
[2] RedHat Enterprise MRG http://www.redhat.com/mrg/realtime/features/
[3] Vun, N.; Hor, H.F.; Chao, J.W.; *Real-Time Enhancements for Embedded Linux*, 14th IEEE International Conference on Parallel and Distributed Systems, 2008. ICPADS '08, Dec. 2008.
[4] Linux Intermediate Queueing Device (IMQ) - http://www.linuximq.net/
[5] Song Kai; Yan Liping, *Improvement of Real-Time Performance of Linux 2.6 Kernel for Embedded Application*, International Forum on Computer Science-Technology and Applications, 2009. IFCSTA '09. vol.2, no., pp.71-74, 25-27 Dec. 2009
[6] Betz, W.; Cereia, M.; Bertolotti, I.C., *Experimental evaluation of the Linux RT Patch for real-time applications*, IEEE Conference on Emerging Technologies & Factory Automation, 2009. ETFA 2009., vol., no., pp.1-4, 22-25 Sept. 2009
[7] O. Mirabella, M. Brischetto, A. Raucea, F Bannò, V. Di Martino, *WTB: a Token Based Wireless Communication over 802.11b*, 2010 International Symposium on Industrial Electronics (ISIE 2010), July 2010.
[8] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications*, The Internet Society, July 2003.
[9] Toby Xu , Ying Cai, *Streaming in MANET: Proactive Link Protection and Receiver-Oriented Adaptation*, 2007 IEEE International Performance, Computing, and Communications Conference , April 2007.
[10] Toby Xu , Ying Cai, *Streaming in MANET: Proactive Link Protection and Receiver-Oriented Adaptation*, 2007 IEEE International Performance, Computing, and Communications Conference, April 2007.