Università degli Studi di Ferrara

DOTTORATO DI RICERCA IN
“SCIENZE DELL'INGEGNERIA”

CICLO XXI

COORDINATORE Prof. Stefano Trillo

Strategies for Multimedia content delivery

Dottorando | **Tutore**

Dott. Andrea Odorizzi | Prof. Gianluca Mazzini

Anni 2006 / 2008

CONTENTS

1. STRATEGIES FOR MULTIMEDIA CONTENT DELIVERY

Today, most of the existing traffic of the Internet will, not conveniently, be treated in a best effort basis.

Of the different traffic types carried, flows with audio and video content (multimedia traffic) have had a substantial growth in the last years, and their added value of *edutainment* motivates both academic and commercial research for solutions on improving the quality of such services.

Multimedia traffic has strict requisites regarding delays, jitter, packet losses and transmission rates. As traffic patterns change may very often and capacity may differ for the various segments in the traffic path, disturbances are likely to happen and the Quality of Service (QoS) being offered to the users may be impaired.

To overcome such problems, there are mechanisms centered on the end systems or on the network; some solutions may use learning techniques to better improve their performance over time.

At the end systems, efficient codification and compression algorithms help to improve the perceived quality of multimedia transmissions, and at the same time can generate flows with the most adequate characteristics (picture size, bit rate, etc.) for each type of terminal.

At the network level, there are mechanisms for differentiating traffic in IP networks, such as the Integrated Services (IntServ) model, the Differentiated Services (DiffServ) model and Multi-Protocol Label Switching (MPLS), where resources are allocated either through the use of static procedures or signaling mechanisms.

However, static procedures do not answer the needs of traffic dynamics, while signaling may involve a significant overhead to follow all traffic profile changes, scalability being a major issue here.

Moreover these static procedures require a dedicated infrastructure that can not be available or can not be realized in the analyzed networks (for example in sensor networks)

Learning algorithms may be included in QoS adapting solutions to improve over time the operational parameters at the network, to change the characteristics of the traffic itself when this is needed and feasible, or both of them.

Such an approach would examine the profile of traffic ingressing in the network and take proper measures to keep the system healthy.

1.1 Overview

The work presented in this thesis faces multimedia content delivery issues in different types of network, such as wired network, the Internet, ad hoc networks and sensor networks.

The term *resource* can refer to different concepts, since, there are several aspects that affect the performances and Quality of Service of the analyzed networks.

The studies discussed in this thesis are mainly referred to the following resources: the available bandwidth, the experimented delay, the experimented jitter and also, in wireless networks, the channel or communication medium to be shared among the networks participants and the energy available for the network nodes.

The aim is to find optimal solutions to allocate these resources in order to improve as much as possible the Quality of Service of the network under exam, in terms of various performance metrics, such as the achievable throughput, the latency and jitter, the energy saving.

The following subsections specify with more details how all these research items have been organized and presented in rest of the thesis. Chapters have been mainly divided accordingly to the type of network the techniques proposed are referred to: overlay network over the Internet, ad hoc and sensor network.

The last Chapter lies outside this subdivision since it refers to an aspect that is common to all the types of networks: the security enhancement of the communications through a shared channel.

Finally some conclusions are reported.

1.1.1 Multimedia Streaming peer-to-peer network

Chapter 2 is devoted to the study of a content delivery paradigm over the Internet.

In particular in this case the resources taken into consideration are the available bandwidth and the experimented delay and jitter.

Overlay networks have been proposed as a way to improve Internet routing, such as through quality of service guarantees to achieve higher-quality streaming media. Previous proposals such as IntServ, DiffServ, and IP Multicast have not seen wide acceptance largely because they require modification of all routers in the network. On the other hand, an overlay network can be incrementally deployed on end-hosts running the overlay protocol software, without cooperation from ISPs. The overlay has no control over how packets are routed in the underlying network between two overlay nodes, but it can control, for example, the sequence of overlay nodes a message traverses before reaching its destination.

This Chapter proposes a new overlay network, focuses on the resilience of this distributed approach and proposes techniques whose aim is to preserve the user perceived Qos also during the peer (overlay network node) failure events.

The proposed strategy merges application layer techniques and overlay routing layer ones in order to takes advantage of both the available information.

1.1.2 Sensor networks delivery strategies

Chapter 3 focuses on delivery paradigm in wireless sensor networks environment.

Geographic routing (also called georouting or position-based routing) is a routing principle that relies on geographic position information. It is mainly proposed for wireless networks and based on the idea that the source sends a message to the geographic location of the destination instead of using the network address.

Geographic routing requires that each node can determine its own location and that the source is aware of the location of the destination. With this information a message can be routed to the destination without knowledge of the network topology or a prior route discovery.

In particular, in order to increase the energy efficiency Chapter 3 proposes a new random geographic routing multisink-multicast approach that does not require any signalation traffic messages. Finally it analyzes the performances and the applicabilty of the proposed routing and delivery protocol.

2. MULTIMEDIA STREAMING PEER-TO-PEER NETWORK

The problem of distributing *live* streaming media content without any dedicated or available infrastructure could lead to peer-to-peer architecture able to establish a generic collaborative environment. The aim of these distributed platforms is to provide an environment in which every peer will be able to share all the media required in the collaborative sessions i.e., synchronous e-learning sessions, video-conferences...

A peer-to-peer approach allows to scale the required resources, i.e., bandwidth, ..., with demand (the amount of interested client) and could allow a performance optimization establishing an overlay network structure. Self-optimizing structure is pivotal in collaborative environment in which participants interact with each other for a long time and in which some of them can leave the session unexpectedly.

The greatest challenge of a peer-to-peer overlay network environment that deliver real time multimedia content is to increase the content distribution resiliency against peer transience.
In the following a distributed approach, named P3P,[1][2], is discussed. P3P is based on tree-structured (therefore Peer tree Peer) overlay network and provides a management protocol that allows structure optimization and also peer failure management.

2.1 P3P in brief

The target scenario of the discussed real time collaborative environment are made by participants that would establish a multimedia session, for example, from a *foreign* network i.e., a network of an organization which is not the one in which the participant "lives". In addition, collaborative session could be infrequent and irregular and could not justify charges related to a dedicate infrastructure that should be sized to serve the maximum amount of client.

Due to the lack of wide spread support for IP Multicast, the simultaneously distribution of the same multimedia content to a potentially large group of client requires a set of resources that could not be commonly available by everyone.

Today overlay multicast can be an alternative to IP multicast [3], [4], [5], [6], [7]. An overlay network is a computer network which is built on top of another network. Nodes in the overlay can be thought of as being connected by virtual or logical links, each of which corresponds to a path, perhaps through many physical links, in the underlying network. The overlay has no control over how packets are routed in the underlying network between two overlay nodes, but it can control, for example, the sequence of overlay nodes a message traverses before reaching its destination

A P2P based overlay approach avoids single participant resource limitation by scaling the request resources with demands. Every multimedia content is produced by a single Peer and has to be delivered to all the other ones, hence establishing a multicasting tree structure is an obvious choice also used in many overlay multicast structures [4], [5].

Tree structure can bound the resources requested to each single peer but introduces a great limitation: every peer is the root node of its own subtree and each single peer failure could compromise the service delivery to a potentially large subset of peer.

A simple but effective solution is to use network path redundancy; this possibility becomes very attractive with the introduction of Multiple Description Coding, [8], that merges network path redundancy with data one.The CoopNet project, [9], developed by Microsoft exploits this strategies.

Multiple description coding is a method of encoding an audio and/or video signal into separate streams, or descriptions, such that any subset of these descriptions can be received and decoded. The distortion with respect to the original signal is commensurate with the number of descriptions received; i.e., the more descriptions received, the lower the distortion and the higher the quality of the reconstructed signal. P3P establishes a multicasting tree for each description of the delivered multimedia content.

2.2 State of Arts

Following the classification of [10] many overlay multicast projects can be classified into two catalogs according to the structure: end-to-end overlay and proxy-based overlay. In end-to-end overlay, every member in the multicasting group shares the responsibility to forward data to other members. End hosts self-organize into a multicasting tree. These end hosts are called multicast nodes in this case. Narada [4], Yoid [3] and ALMI [7] are some examples of such structure. Using the proxy-based overlay structure, Scattercast [5] and Overcast [6] form a *hierarchy* structure compared to end-to-end overlay. The multicasting service is fulfilled with the help of proxy-based multicast nodes, which can duplicate data and forward data to end hosts with predefined routing algorithm. The concept of multicast node is slightly different in two structures, but in both cases, a multicast node is defined as a member in the multicasting tree.

In the following four projects that are representative of overlay multicast are briefly introduced: Scattercast, Overcast, Narada and ALMI. The design objectives are different from each other which lead to differences in their design approaches and properties in many aspects.

Narada and Scattercast intend to minimizing delay for each member; however, Overcast maximizes available bandwidth for each member. ALMI tries to minimize the system cost, where the cost of each link is defined as the round-trip delay between group members.

Narada and Scattercast use a mesh-first approach, that is, group members are connected in a mesh first and then the multicast tree is built on top of the mesh. Overcast and ALMI use a direct approach. In this case, the step to build the mesh is bypassed and the multicast tree is formed directly.

Overcast and Scattercast claim good scalability, the other two only can serve dozens of members.

2.2.1 Scattercast

Scattercast is an overlay architecture to carry out multicasting by incorporating proxy-based multicast nodes.
When a new node joins a multicasting session, it bootstraps itself via a well-known list of rendezvous points and then relies on the gossip-style discovery algorithm to locate other members. When the new node encounters other members who have already been in the session, it selects some of them as its neighbors if they satisfy the degree constrains.
The initial mesh is randomly formed. When performing optimization, latency is the primary metric considered. Member decides to accept others as neighbors or to change neighbors according to a predefined cost function and threshold. After the mesh is stable, a distance vector routing protocol [11] is running on the top of the mesh, taking latency between neighbors as its routing metric.

2.2.2 Narada

Narada intends to serve small size and sparse groups such as audio-video conferencing and virtual classrooms.

Narada and Scattercast use the mesh-first approach, but are different in process of mesh optimization. Narada assumes that the new node is able to get a list of group members by an out-of-band mechanism. It randomly chooses some members as its neighbor if those members would not exceed their maximum degree constraint. It exchanges messages with its neighbors to learn other members.

Every member periodically evaluates the utility of adding a link and deleting existing links to decide the further optimization.

The way of building a data delivery tree and routing is the same as Scattercast.

2.2.3 ALMI

The design goal of ALMI is to minimize the cost of the system, that is, the cost of the distribution tree. The distribution tree in ALMI is formed as a Steiner Minimum Tree, *SMT*, where the cost of each link is the latency of the link.

The operations in ALMI greatly depend on the *central control server*, which collects the latency information and calculates a *SMT* to be sent back to all members.

Such a centralized control approach simplifies the routing problem compared to a distributed approach, however, it only works for a small communication group and also suffers from the single-point failure problem.

2.2.4 Overcast

Overcast directly builds multicast trees to maximize bandwidth to the source for all members at the expense of the potential to increasing of delay.

When a newly initialized node join the group, it contacts the root as the first step. Then the new node re-evaluates its position and tries to locate itself further away from the root without sacrificing bandwidth back to the root.

In each re-evaluation round, the node evaluates the bandwidth to the current parents and the bandwidth to the children of the current parents. If the bandwidth through any of the children is about as high as the current bandwidth, that child will be chosen as the current parent.

With such a strategy, the node will be located as far from the root as possible without reducing bandwidth.

2.2.5 CoopNet

All the previous projects are not addressed to realize a resilient peer-to-peer architecture and marginally consider the problem of a peer failure or the possibility of join to the multimedia session for a very short time compared to the other peer session duration.
Multimedia description coding offers the possibility of receive different descriptions of the same multimedia content fron different sources hence it allows the realization of a resilient distributed architecture in which a peer failure does not compromise the service delivery. CoopNet projects, [9], exploits this possibility; it organize the peers in different distribution trees, one for each description.

The CoopNet distributed architecture is managed by a *central server* that organizes peer tree evolution, decides the MDC parameters and also can serve a large amount of client.
CoopNet architecture is designed to deliver the same multimedia content to a very large set of client (often with a very irregular connection time) and the aim of the optional distributed peer-to-peer network is to aid this central server.

2.3 Multiple Description Coding

Multiple description coding, *MDC* is a method of encoding an audio and/or video signal into $M > 1$ separate streams, or descriptions, such that any subset of these descriptions can be received and decoded.

The distortion with respect to the original signal is commensurate with the number of descriptions received, i.e., the more descriptions received, the lower the distortion and the higher the quality of the reconstructed signal.
This differs from layered coding in that in MDC every subset of descriptions must be decodable, where as in layered coding only a nested sequence of subsets must be decodable.

Many multiple description coding schemes have been investigated over the years. [8] contains a simple overview. A particularly efficient and practical system is based on layered audio or video coding, Reed-Solomon coding, and priority encoded transmission.

In such a system the audio and/or video signal is partitioned into groups of frames (GOFs), each group having a duration of $T = 1$ second or so, for example. Each GOF is then independently encoded, error protected, and packetized into M packets, as shown in Figure 2.1. Both layered coding and Forward Error Correction (FEC) are building blocks for MDC.

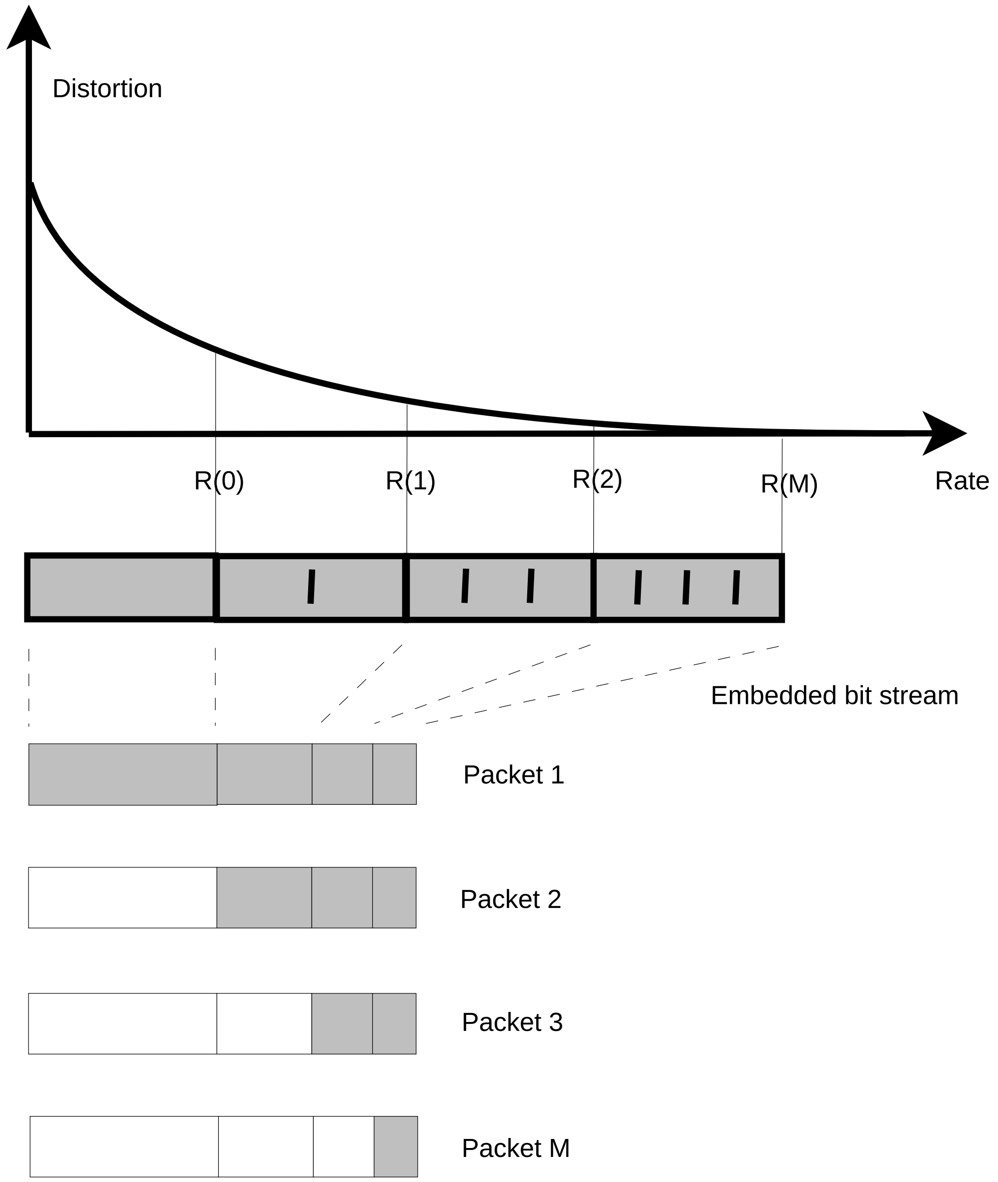


Fig. 2.1: Priority encoding packetization of a group of frames (GOF). The source bits in the range $[R_{i-1}, R_i)$ are mapped to i source blocks and protected with M_i FEC blocks. Any m out of M packets can recover the initial R_m bits of the bit stream for the GOF

Layered coding is used by MDC to prioritize the streaming data. The bits from a GOF are sorted in a decreasing order of importance (where importance is quantified as the bit contribution towards reducing signal distortion) to form an embedded bit stream. For example, bits between R_0 and R_1 are more important than the subsequent bits in the embedded stream in Figure 2.1. Forward Error Correction (FEC), such as Reed-Solomon encoding, is then used to protect data units to different extents depending on their importance.

M descriptions can accommodate up to M priority levels for a GOF. If any $m \leq M$ packets are received, then the initial R_m bits of the bit stream for the GOF can be recovered, resulting in distortion $D(R_m)$, where $0 = R_0 \leq R_1 \leq \ldots \leq R_m$ and consequently $D(R_0) \geq D(R_1) \geq \ldots \geq D(R_m)$. Thus all M packets are equally important; only the number of received packets determines the reconstruction quality of the GOF.

A smart tuning of MDC parameters, i.e. total number of descriptions, GOF durations, . . . , requires, for example, a very accurate estimations of the probability distribution of the amount of received descriptions.

The problem of tuning the parameters of MDC, that is pivotal in order to optimize the perceived quality of service, is beyond the scope of this discussion. Instead of a distortion analysis approach in the following is used an heuristics metric in which the measure of the perceived quality is only the amount of received descriptions: the more descriptions received, the higher the quality of the reconstructed signal.

2.4 Architecture Overview

Following the classification of [10] P3P project can be classified into end-to-end overlay network set; every member in the multicasting group shares the responsibility to forward data to other members and all together they organize theirself into a multicasting tree.

The entities of P3P architecture are Peer and User Agent (UA). The UA is the human front-end that transmits and receives all the multimedia streams; the peer, which is a node of the multicasting tree, act as a proxy from the UA point of view, and serves a small amount of UAs delivering to them the streams received in the multicasting tree, figure 2.2.

Peer and User Agent are not necessary separated and they could collapse in a single object, but dividing them allows to decouple the overlay multicast structure from the specific front-end application. Moreover it allows to take advantage of local IP multicast infrastructure: P3P could be used also in order to connect different multicast network.

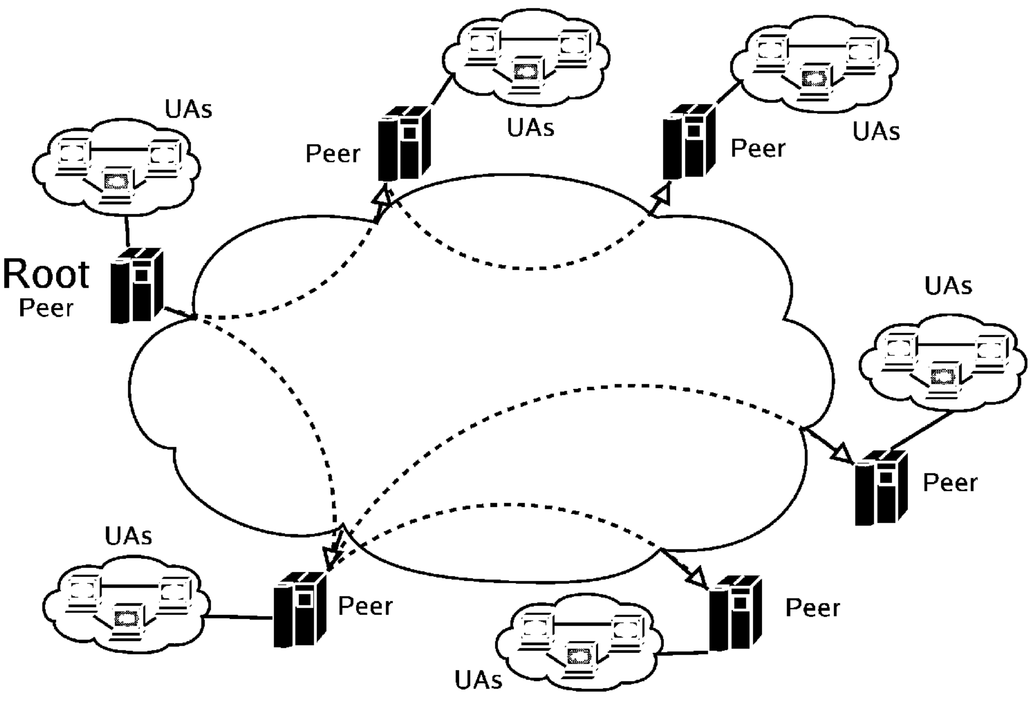


Fig. 2.2: Architecture overview, *all the peers establish an overlay multicasting tree network and deliver or receive multimedia contents to their own UAs*

The mean of the local network peer-UAs interaction environment, the one in which all the UAs are connected to the same peer, is to quickly connect a small amount (2 or 3) of participants without increase the global structure load: all the UA connected to the same peer communicate directly to each other and, if a UA is a stream source, it will send a copy of the captured streams to the peer that can encode them in different description and propagate them through the multicasting tree.

No infrastructure means no rendez-vous server. Following the bit-torrent example a small *.p3p* file could be published over a well known web site or could be sent by mail. The obtained *.p3p* file contains all the information required in order to contact the root peer or another node of at least one multicasting tree.

Each peer that know the information included into *.p3p* file will automatically integrate itself in the overlay multicast structure following the tree management protocol rules. Like in Scattercast, section 2.2.1, nodes bootstrap theirself via a well-known list of rendezvous points and then rely on the gossip-style discovery algorithm to locate other members.

Multiple Description Coding (MDC), [8], allows to merge data and network

path redundancy.
In order to avoid lack of services due to peer failure events network path redundancy is pivotal : If the streaming content is encoded using MDC, different description will be distributed over different distribution trees hence peer failure does not always compromise all the description received from a potentially large sub-set of peer, because the same set of peer, in another distribution tree, could not be positioned into a failed node subtree.

In Figure 2.3 is depicted a very simple situation in which root peer, the one that introduces multimedia content into the multicasting tree, distributes two description through two different trees. The effect of the a-peer failure event is reduced to a simple degradation of the perceived quality at nodes b,c,e and does not lead to a complete out of service for this particular subset of node.

Different overlay distribution trees have to be as *orthogonal* as possible, i.e. the higher is the amount of shared overlay network links, the higher is the probability of a node sub-set out of service or heavy quality degradation event.

Furthermore, adopting MDC, P3P can realize a differentiated class of service: peers with poor available bandwidth resources can join only a reduced amount of distribution tree hence receive a reduced amount of description.

MDC can also aid peers to manage starvation situation: reducing the amount of forwarded descriptions a peer can deliver the reduced set of description to an enlarged amount of child node.

Each peer delivers as much description copy as allowed by its available bandwidth; during tree optimization phase could be convenient to temporarily accept lots of *child nodes* (for example in order to aid nodes that will be relocated into the tree structure) and this strategy could be realized by reducing the amount of description copy delivered to the current child nodes.

2.5 Tree Management

In this section the problem of constructing and maintaining a single overlay distribution tree is discussed.

Unlike [9], in which each description or distribution tree is organized by the central server, in P3P each tree structure grows and tries to optimize itself, at first, through a greedy algorithm and, hence, using a distributed local search optimization.
In the following discussion all the nodes participate and contribute to resources sharing until the end of the session or until a failure event.

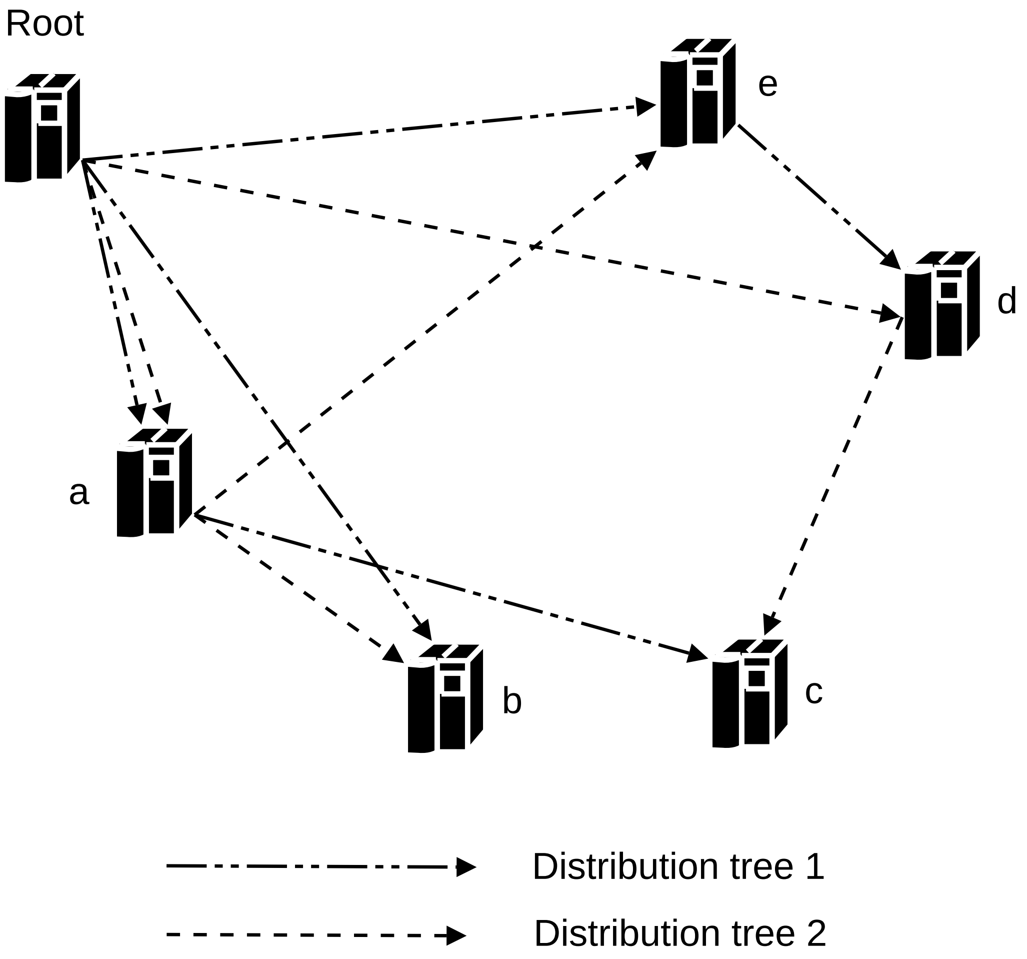


Fig. 2.3: Distribution Trees, *Root node produces 2 descriptions of the same multimedia content that are delivered through two different overlay distribution trees*

2.5.1 Parent and Child Metric

Unlike [3], [4], [5], [6], [7] that decide the overlay network evolution evaluating a single parameter (bandwidth, delay, ...) measured by each peer, P3P adopts two evaluation metrics in order to decide which peer could be a good parent node or which peer could be a good child node. Each peer takes its decisions comparing the neighbour peer attributes and/or the attributes of peer that is interacting with it.

- **Parent metric** is used in order to decide which peer could be the better parent node in the overlay multicasting tree and is based on
  - the evaluation of peer *level*, i.e.,the amount of intermediate nodes between the tree root and the candidate parent node
  - the measured *round trip time*
  - the the amount of common *ancestor node*,i.e., node between the tree root and the candidate parent node, between different distribution tree.
- **Child metric** is used in order to decide if a peer could be accepted as child node in the everlay multicasting tree compared to the actual child peers; it is based on
  - the outgoing bandwidth published by child candidate peers
  - the current outgoing available bandwidth of the same node (child node candidata can be a root of an overlay subtree)
  - the amount of common *ancestor node*
  - the measured round trip time

Tentatively a parent peer is better than another one if its level is lower than the other one level, instead a child node is better than another one if its outgoing bandwidth is greater than the one of the other node.

Morevoer the metrics could and should be tuned in order to advantage the influence of some parameter. Nevertheless bandwidth evaluation are pivotal in order to reach the goals listed in section 2.5.2

2.5.2 Tree management goals

The goals of the discussed single tree management algorithm are the following:

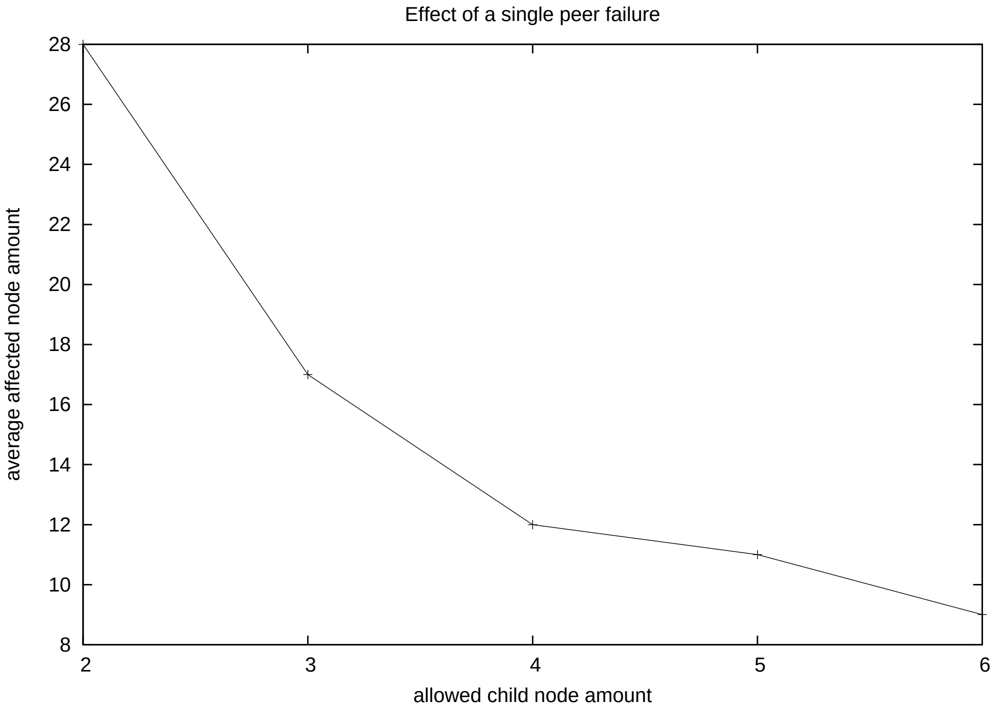


Fig. 2.4: Average amount of node affected by a single peer failure in a 255 nodes single distribution tree

- **Bushy trees:** P3P tries to reduce the height of each tree structure i.e., it minimizes the amount of intermediate nodes between the root node and the further away nodes. Shortness would minimize the introduced overlay multi-hop delay and than minimizes the probability of sub-tree disruption due to peer failure event.
  In figure 2.4 is depicted the effect of enlarging the child node amount: it can reduce the average number of nodes affected by single peer failure event.

  Each node selects its own child nodes evaluating the *Child metric* values of all current child nodes, if at least one exists, and evaluating the metric value if the new incoming child node candidate.

  Each node that searches a parent peer, sorts the peers that could become his own parent node by evaluating *Parent metric* over all the parent node candidates.

  Nodes with large outgoing bandwidth, hence, potentially lots of children, will obtain a low level in the distribution tree therefore each overlay tree becomes as bushy as possible without external or centralized coordination.

- **Soft handover:** If a parent node that delivers multimedia content to its maximum amount of child node receives a join request sent by an interesting child candidate node, i.e., a peer that exhibits a child metric value greater than at least one of the current child node value, the current child node with the lower metric has to be relocated into the P3P structure.

  In order to avoid services interruption, descriptions directed to each child node has not to be stopped until the rejected child node finds a new parent one.

- **Gossip-style signaling:** In order to reduce the signalation traffic discovery messages i.e., messages used to locate system resources, are carried following the overlay network established paths: each node forwards the incoming discovery message over all its established interface (parent, child *a*, child *b*,...) but the one from which the message has been received. All the other type of signaling message involves only two peers that communicate directly.

- **Smart optimization:** P3P sessions could be very long and the underlying network properties could significantly change during each multimedia session. In order to follow the underlying network status, if a peer register a *stable* situation, i.e. does not receive signaling message for a significantly time period, it will try to optimize its own position into a distribution tree.

- **Smart failure recovery:** After a peer failure at least one sub-tree register a perceived quality decreasing. In order to avoid a discovery flooding, only child nodes of the fault peer have to search a new parent node; other nodes in the orphaned sub-tree wait for an ancestor node driven reconnection.

2.5.3 Tree management protocol Client-side : join single multicasting tree

In order to join to the P3P platform in the behavior of the joining peer could be identified five different macro states, figure 2.5, each one related to a specific interaction between the peer and the overall P3P structure.

- **Lock**: In order to avoid loop creation during simultaneous sub-tree migrations each node that has to move from a parent node to another one has to prevent incoming join requests in its own sub-tree.

- **Discovery**: the aim of discovery phase is to aid node that tries to join to the structure has to know which is the set of possible accommodation into the distribution tree and, moreover, which are the nodes that constitute the multicasting tree in order to chose the best available position.

- **Join**: During the Join phase resources have to be allocated from nodes in the P3P structure to the joining peer. After Discovery conclusion each node selects the better parent candidate node evaluating parent metric and, finally, try to join the structure.

- **Unlock**: Unlock phase is used in order to renew the possibility of external nodes to join into the locked sub-tree.

- **Sleep**: Sleep is the macro state in which the peer does not perform any tree management procedure in order to modify its own position into the distribution tree structure.

The sequence of macro state in figure 2.5 performs the basic flow used by peers in order to join a multicasting tree.

Nevertheless the same flow with some obvious differences is used in order to optimize node positions into the multicasting tree and, also, in order to perform sub-tree lock and unlock procedures originated by ancestor node.

In figure 2.6 these three flows are depicted:

- **Basic flow** is the red one. It is performed in order to find at least a new parent node. For example nodes that start a new session perform the red flow. Another example happens when a parent node decides to relocate one of its child nodes: removed child nodes perform the red flow. At last red cycle is performed when peer detects the parent node failure event.

- **Optimization flow** is the green one. It is performed in order to obtain a better position into the multicasting tree structure. After sleeping for a specified time period, $timeout_{flat}$, each node starts an optimization procedure: it starts to lock its own subtree and then performs the *discovery* procedure hence, if possible, it performs the *join* procedure.

  Unlike basic flow if the discovery procedure does not find any parent node, if discovery procedure finds only parent candidate nodes with parent metric values smaller than the current parent one or if the join procedure does not complete with success, the optimization flow continues with the unlock procedure and does not contine to search a parent node executing another **discovery**.

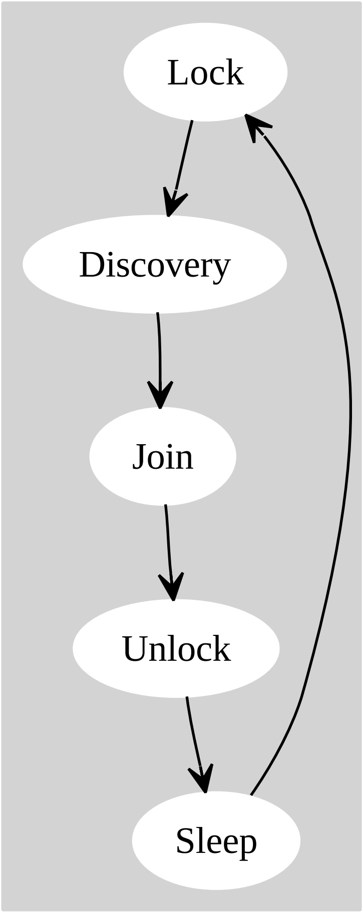


Fig. 2.5: Macro state sequence of the basic execution cycle performed by a node in order to join the P3P structure

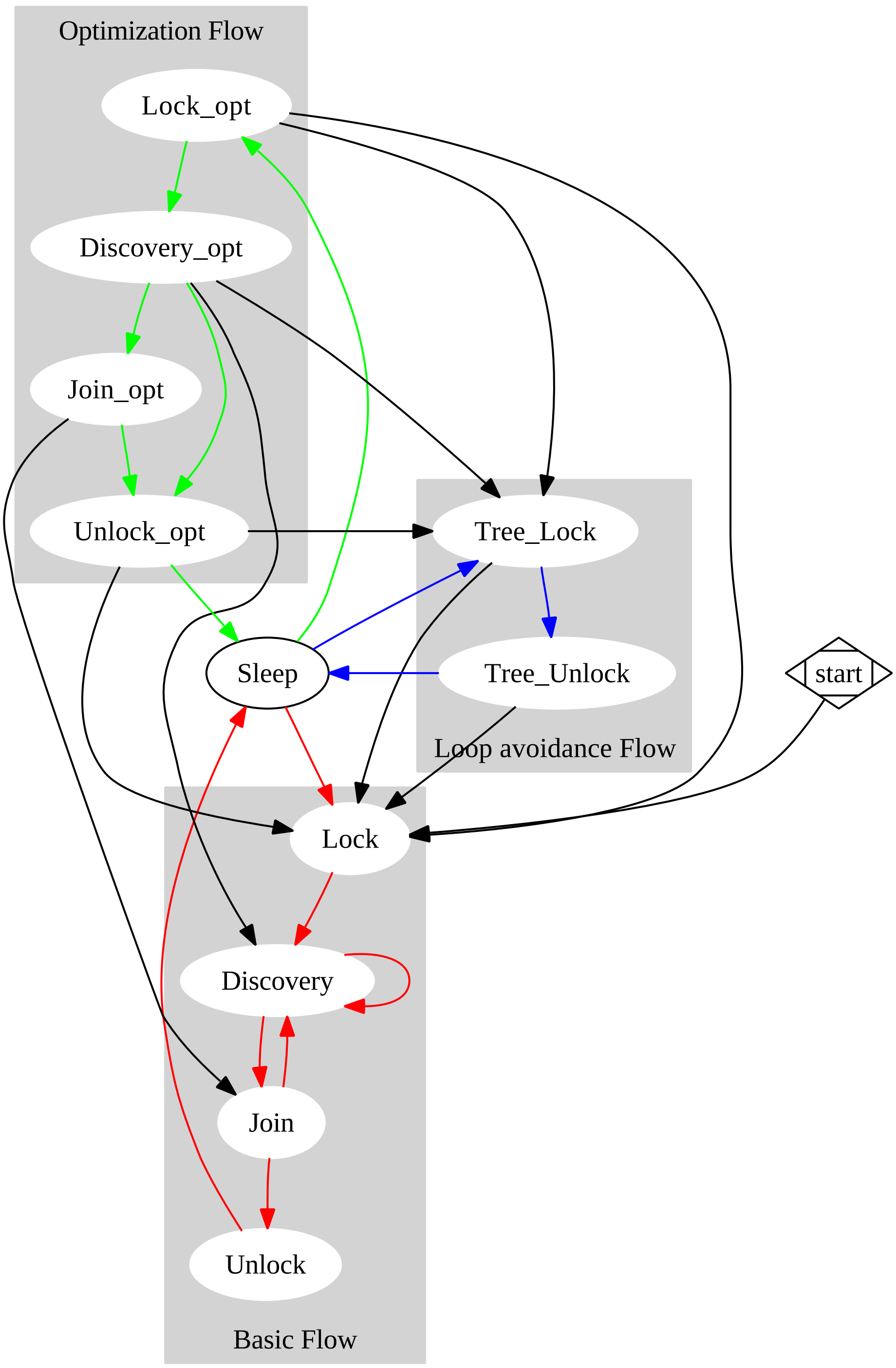


Fig. 2.6: State diagram of the execution flow performed by the client side of a node in order to join or optimize its position into the P3P structure

Vice versa the execution flow migrates into the basic one, the join procedure is completed and a better position into the distribution tree is obtained.

- **Loop avoidance flow** is the blue one. It is a simplified flow driven by an ancestor peer that decides to lock (and unlock) its own subtree.

Each flow has got a priority and the execution flow can not migrates from flows with higher priority to ones with smaller one. In detail the basic flow has a priority higher than the loop avoidance one, whose priority is also higher than the optimization flow.

For example nodes that try to optimize their position and receive a *lock request* from their parent node leave the green flow and start the blue one, but nodes that have to find, in the red cycle, a parent node ignore the incoming *lock request.*

A much detailed executing flow graph is depicted in figure 2.7 in which the elements of figure 2.6 is not grouped by flow

2.5.4 Tree Management Protocol Server-side: manage new peer

In order to manage the distribution tree join request of the incoming peers, the behavior of each peer that receives and manages the join request message could be simplified in the sequence of the following macro states:

- **Join**: In this macro state a new *join request* is managed. The peer evaluates, with the aid of the *child metric*, if the incoming child node candidates could be accepted or not. If the incoming node can not be accepted, i.e., peer outgoing resources are not available or dedicated to child nodes with greater *child metric*, peer stops the join procedure and returns in the sleeping macro state.
- **WaitingConfirm**: In this macro state peers that have accepted a *join request* are waiting for *join confirm* message from the incoming peer. If the confirm is negative peers will return in the sleeping macro state.
- **Allocate**: In this state peers that have received a positive *join confirm message* allocate the outgoing resources requested by the incoming child nodes.
- **Reject**: If the peer outgoing resources are completely allocated but the incoming child candidate peer metric values is greater than at least one of the current child node ones, the new incoming child will be frozen

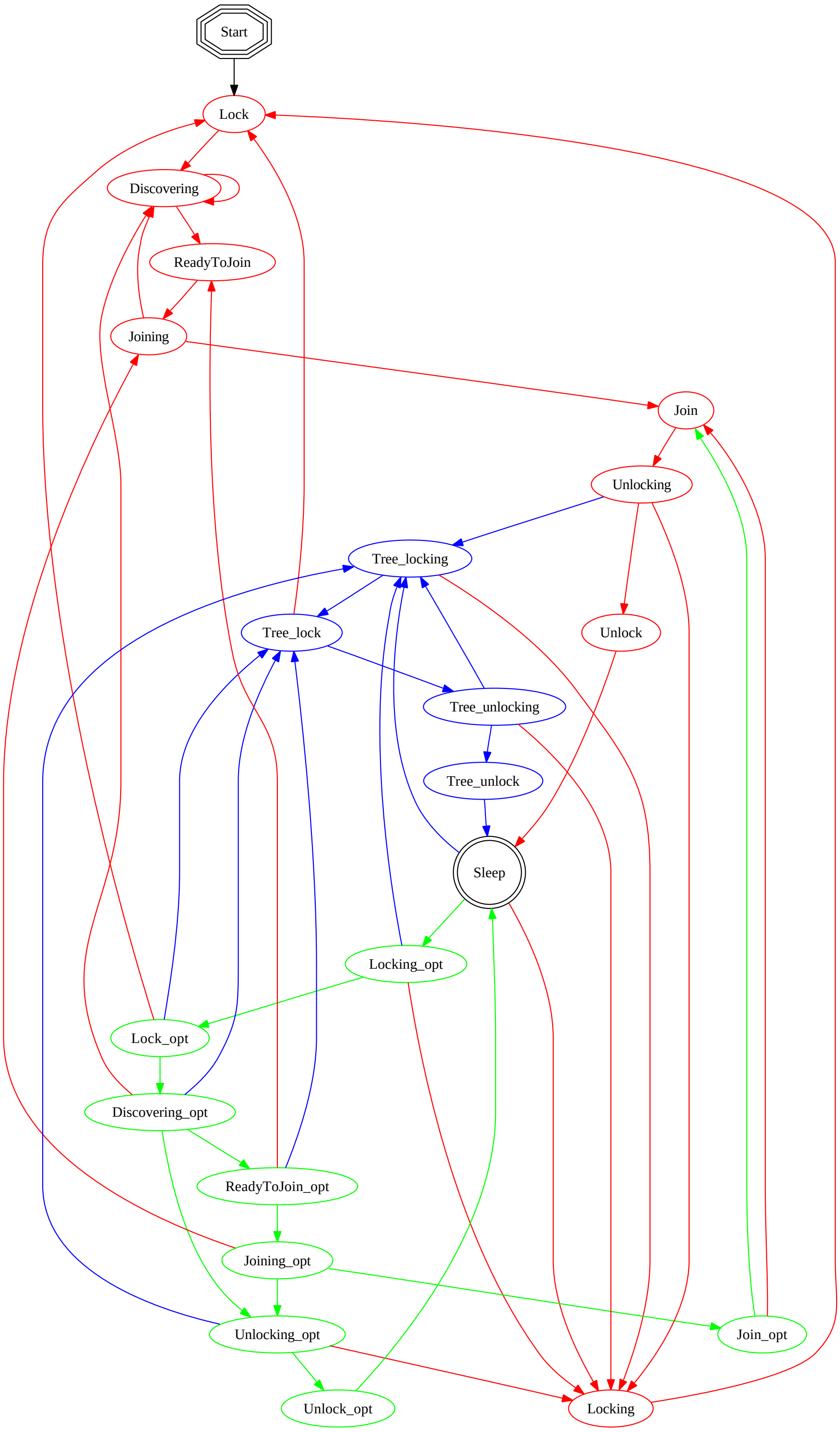


Fig. 2.7: State diagram of the execution flow performed by the client side of a node in order to join or optimize its position into the P3P structure

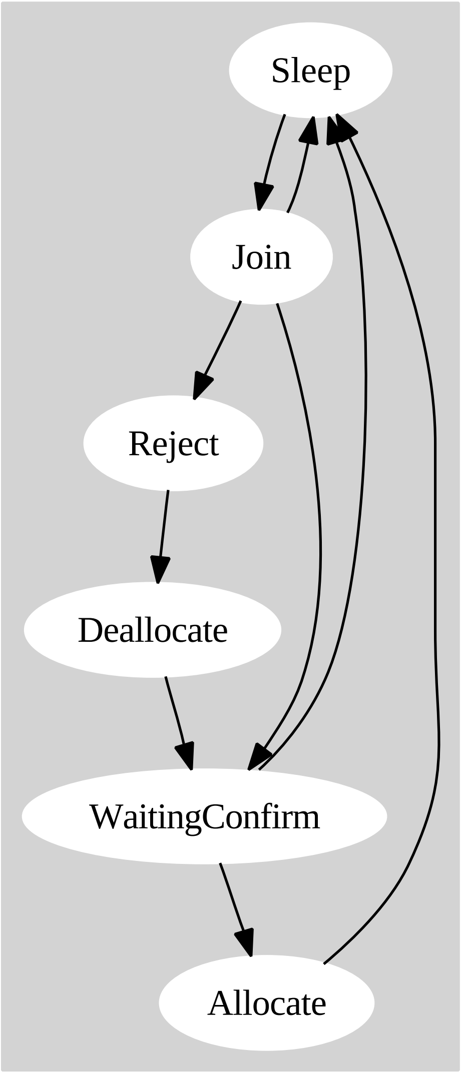


Fig. 2.8: Macro state sequence of the basic execution cycle performed by a node in order to accept a joining request for the P3P structure

until outgoing resourced will be relocated. A reject request is sent to the worst child node (the one with the smaller child metric) and peer still wait for the reject reply.

- **Deallocate**: In this state peers deallocate the outgoing resources dedicated to their worst child peers and reply to the incoming child in order to complete the previous join procedure.
- **Sleep**: In this state peers wait for incoming join requests.

The sequence of macro state in figure 2.8 constitutes the basic execution flow exploited in order to accept a new peer in each distribution tree structure.

Nodes in the P3P structure interact not only with the incoming peer but also with neoghbor nodes in the overlays. The execution flow depicted in in figure 2.9 is grouped into the three sub-flows that have to be managed by a peer in order to accept or not an incoming child node and takes into account of neighbor interactions.

- **Join Management Flow** Is the execution flow performed if the peer receives a *join request*. Green sub-flow is performed if the node can allocate immediately resources for the incoming peer otherwise the red one is performed. The difference between red sub-flow and green one is that, in order to allocate resources, the peer has to remove one, the worst, of its current child nodes. The worst child node is the one with the smaller *child metric*.

  At the end of resources deallocation join procedure is performed: incoming peer, that is waiting for the definitive join reply, could continue the join procedure with an handshake and peer resources are reallocated.

  The join procedure can manage only one child node at a time. Brown flow takes into account the possibility of receive more than one requests. Like in the reject phase, after receiving multiple requests, incoming nodes are *frozen* in order to complete the current join procedure. If multiple frozen child nodes exist the new join procedure involves the node with the greatest child metric.

- **Deallocate Resource Flow** Is the flow performed at the reception of an unrequested *Reject Reply* (for example at the end of a successful optimization procedure performed by a child node). The effect of this event during all the execution flows is the same: resource deallocation. This event also affects the join execution flow (see purple macro state transitions in figure 2.9).

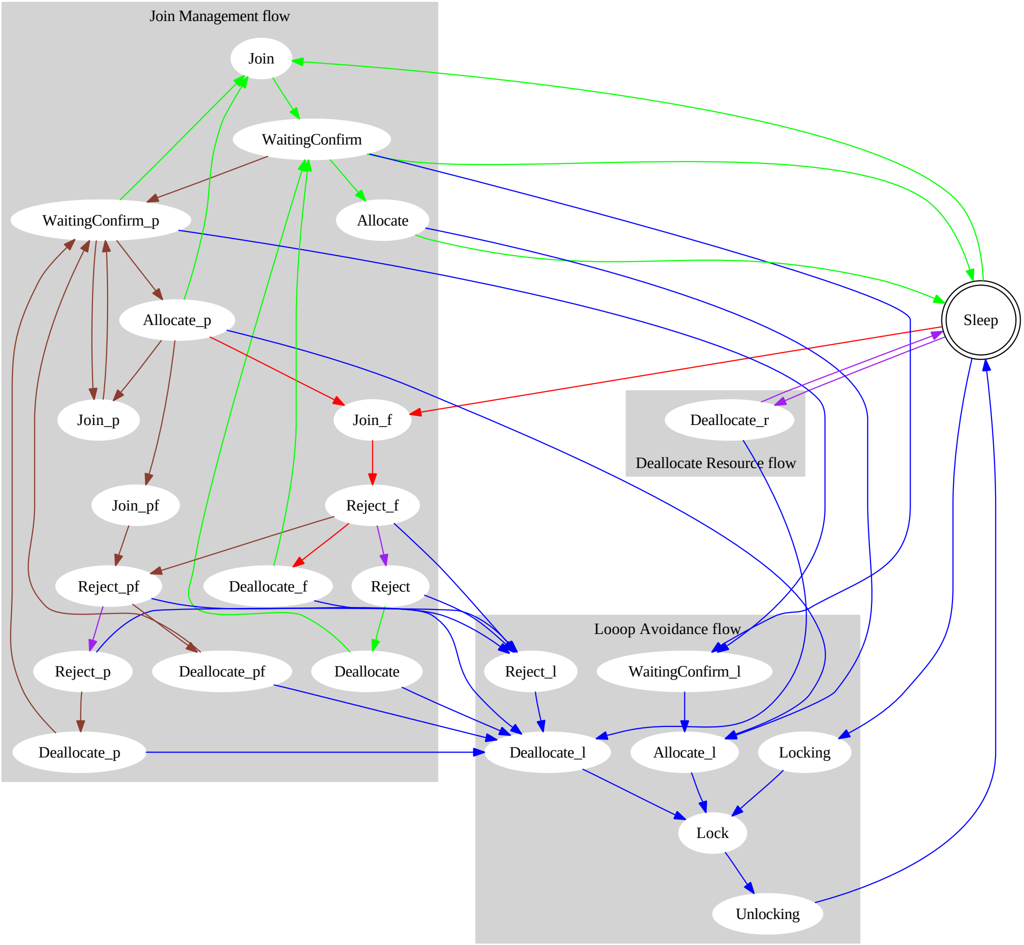


Fig. 2.9: State diagram of the execution flow performed by a node in order to accept and manage a join request message

- **Loop Avoidance Flow** Is the flow performed in order to block the peer sub-tree and it is triggered by the reception of a lock request from parent node. This event affects all the other execution flow: see blue transition in 2.9. During this transition all the frozen incoming child nodes will be rejected by the peer and only the current incoming child peer or the current reject phase will continue their executions in the locking Flow.

  Unlocking sub-tree procedure is also triggered by a parent node request.

Loop Avoidance flow is performed with an higher priority than Join Management flow. Therefore peers in the join flow can migrate in the loop avoidance flow and peers in the loop avoidance execution flow ignore all the incoming join request. Deallocation flow is performed with the highest priority:

resource deallocation has to be completed as soon as possible in order to relocate immediately the resources.

A much clear execution flow graph is depicted in picture 2.10 in which the element in figure 2.9 are not grouped by flow.

2.5.5 Tree management Protocol: automatic recovery

Both client-side and server side execution flow rely on the idea that each incoming peer, ech child peer, each parent candidate peer and each parent peer are running and can reply to the analyzed peer messages.

In order to take into account of peer failure the following recovery procedures are performed at each $timeout_{neighbors}$:

2.5.5.1 Neighbor check

- **CheckParent**: Peer sends a keep-alive message to the parent and checks the last reply timestamp. If the last reply timestamp is outdated (older than $timeout_{dead}$ milliseconds) a pparent failure event is supposed and a new parent search is triggered.

- **CheckChildren**: Peer checks if the last ingoing keep-alive message of every children is outdated. If so the dedicated children resources are immediately deallocated.

Previous check are based on the following message handshake that is triggered by the CheckParent evaluation.

- **ChildAlive message** If peer receives an incoming *alive request* from child node it replies and waits for the confirm message

- **ChildConfirm message** If peer receives the *alive confirm* message updates the calculated RTT between it and its own child node.

- **ParentReply message** If peer receives a *alive reply* from parent node it sends an alive confirm message and update the calculated RTT between it and its own parent node.

Performing these recovery strategies outgoing resources can be relocated, peer failure can be managed and new parent discovery will be forced.

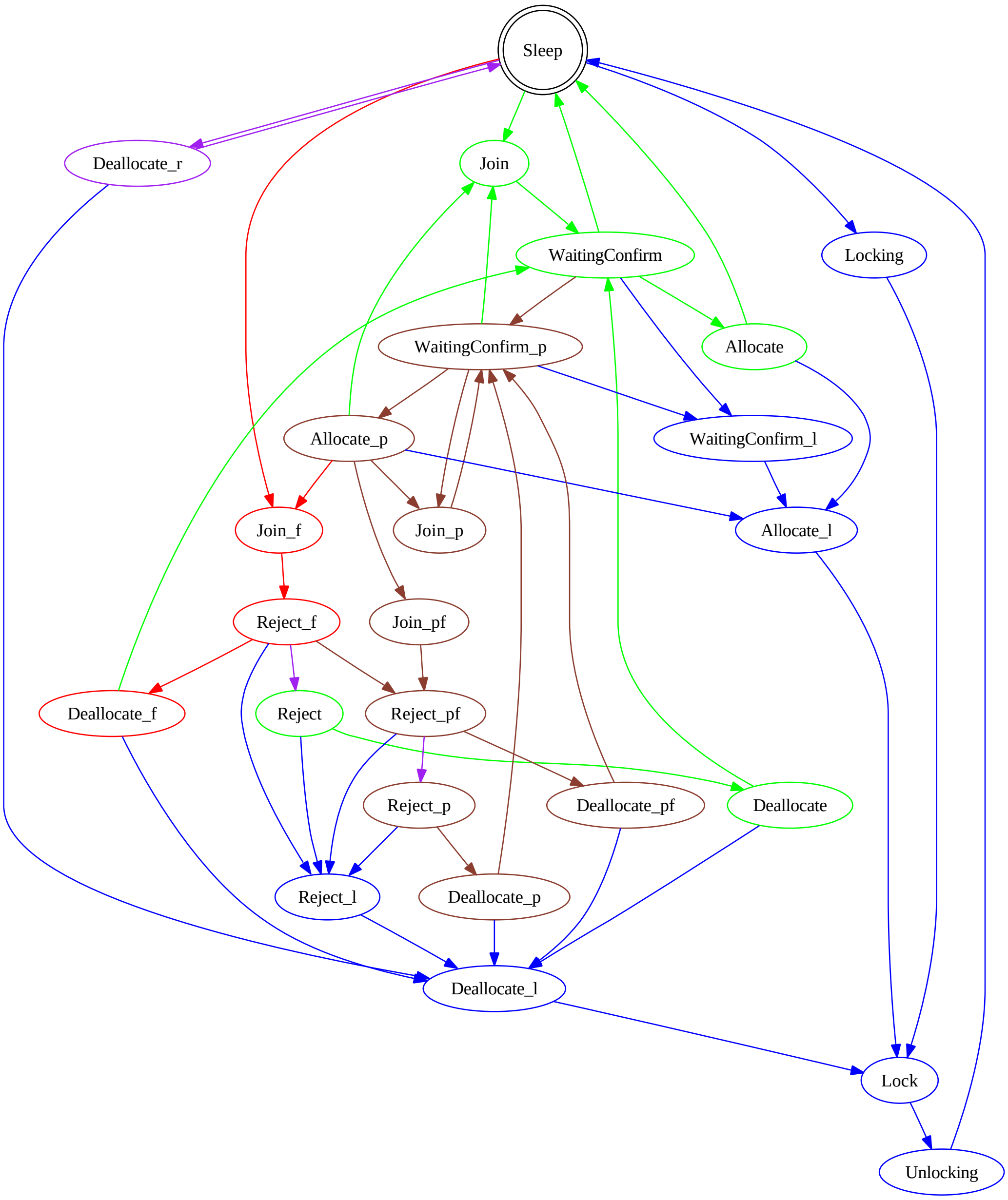


Fig. 2.10: State diagram of the execution flow performed by a node in order to accept and manage join request message

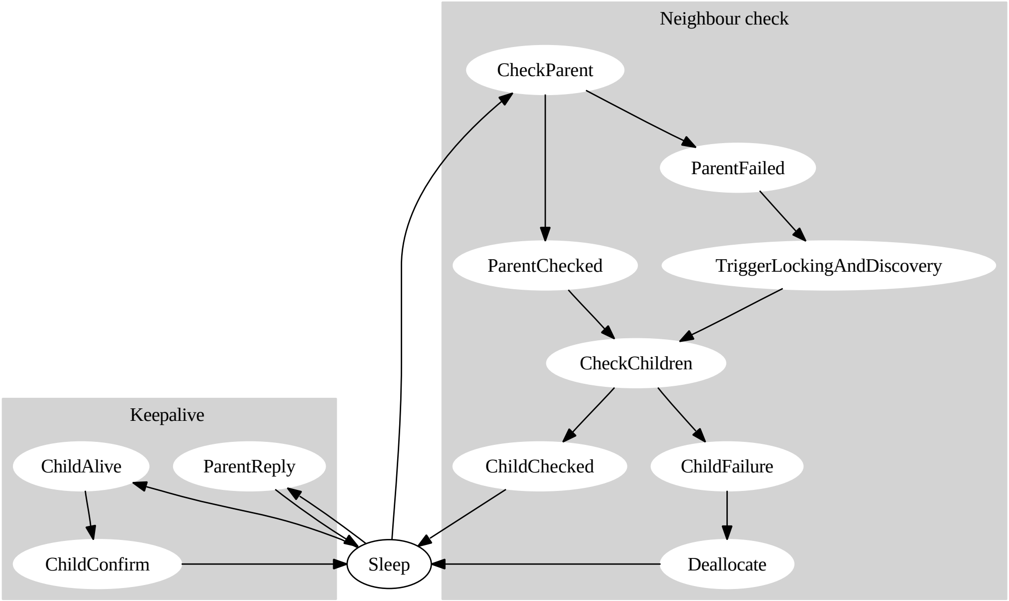


Fig. 2.11: Recovery execution flow

2.5.6 Tree management Protocol: procedures in detail

2.5.6.1 Neighbor characterization

Each node updates a *image* of the overlay P3P network based on the other peers received messages.

In this image each neighbor has to be identified within a particular state that identifies the relation in the P3P structure between the peer and the neighbor node.
Peer behavior decisions are taken based on this states but also on the other data stored in the P3P image,i.e., all the remote peer published attributes, and on current incoming messages.

Remote peers are classified into the following states:

- **unknown** a peer that cannot be identified as a neighbor
- **probed** a peer that has replied to a discovery request. It is not positioned in the peer sub-tree
- **parent candidate** the best probed peer (the one with the higher parent metric) at the $timeout_{discovery}$ expiration; The join request messages are sent to the parent candidate node.

- **parent** the peer which has accepted the join request. Only one parent node per tree should exist. Parent node allocates a fraction of its outgoing resources and pushes the received distribution tree contents to the peer.

- **bad parent** a parent node that would reallocate its outgoing resources from the examined node to another peer; it periodically sends a reject request to the analyzed node but it will continue to push the tree distribution content until the peer replies to the request with a reject reply message.

- **child** node that receive a copy of the distribution tree content and for which a fraction of outgoing resources are allocated.

- **frozen child** a possible child node whose join request could not to be immediately accepted. A node could be identified as frozen child because parent node has to complete a reject procedure or another join procedure. Neither resources are allocated for frozen child nodes until they become child node.

- **joining child** a node that is going to become a child one. It will become a child node only after the join confirm message reception. No resources are allocated until joining ones become child node.

- **rejected child** a child node that has to be removed in order to relocate its dedicated outgoing resources. The distribution tree content has to be pushed to it until a reject reply message is received.

2.5.6.2 Discovery procedure

Discovery procedure are not reliable and requires the execution of lock procedure before it can be performed. In figure 2.12 node Q sends a *discovery request* message to its own current parent node or, if it does not exist, to the distribution tree root node and waits for the $timeout_{discovery}$ expiration. The current parent node could be both *parent* and *bad parent* node.

Node that receives the request forwards the message to all the neighbor nodes but the one from which the request is received. Moreover nodes that are locking their own subtree do not propagate the discovery request to their child nodes.

All peers that receive a discovery message forward it, but reply to Q only if *child metric* of Q is larger than at least one of the child node metric evaluated values or if they can accommodate immediately another child node.

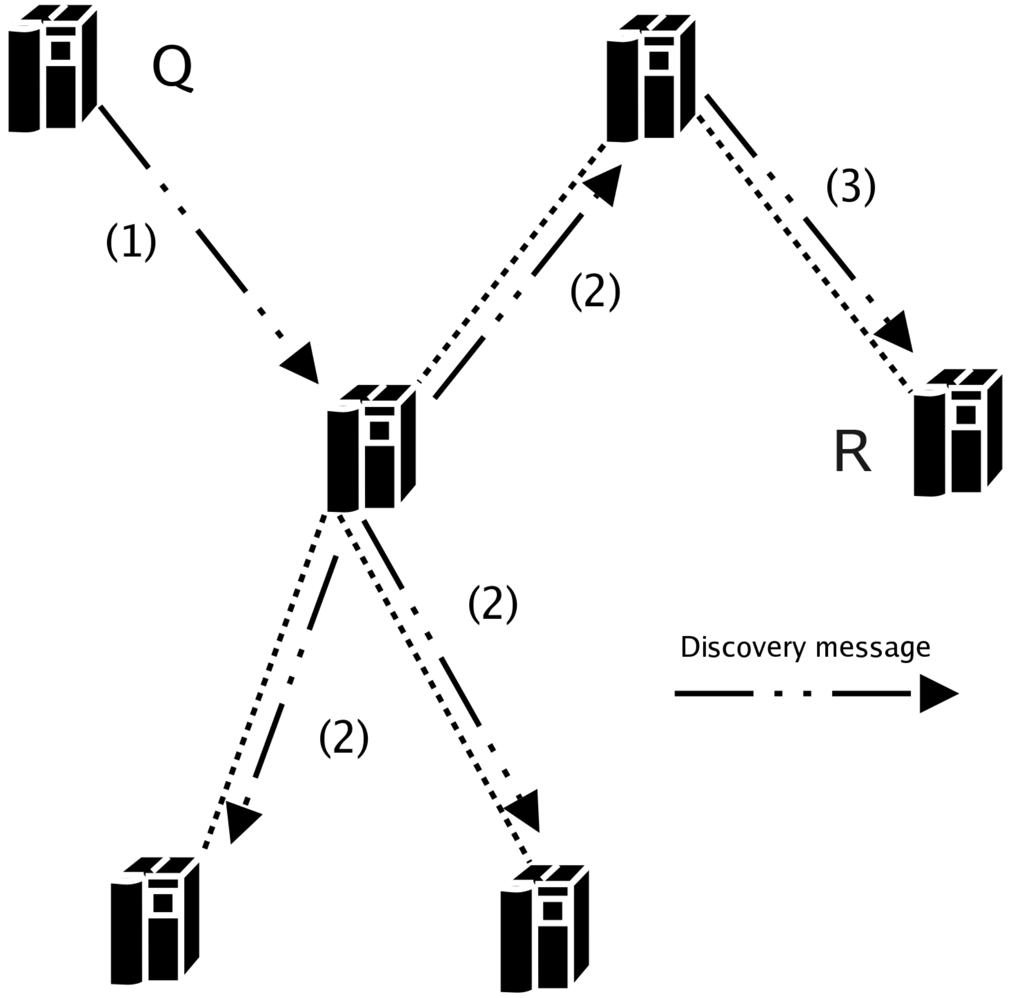


Fig. 2.12: Discovery phase, *Q sends the discovery message to a known Peer, (1), that forwards it to the child nodes and to the parent node, (2), that forward the message too, (3).*

In order to reduce signalation traffic if a peer replies to Q it does not forward the discovery request message to its child nodes: In each subtree child node parent metric values are always smaller than the parent node parent metric one.

Discovery reply message is sent directly to Q and not use the overlay established path. If Q receives the reply messages it will set the state of the remote peer to *probed node* and it will learn that the sender is not in its sub-tree.

At the $timeout_{discovery}$ expiration Q evaluates the set of probed node and elects the best one, the one with the larger parent metric value, as *parent candidate* node.

Discovery reply is also used to measure the path delay between peers:

1. Discovery reply is sent at time t_{reply} (saved by the remote peer).
2. Q receives the discovery reply message at time t_{r2} and sends back a *RTT request* message to the remote peer;
3. Remote peer receives the *RTT request* message at time t_{r3}; it estimates RTT as $RTT_{measured} = t_{r3} - t_{reply}$ and sends a *RTT Reply* message to Q.
4. At time t_{r4} Q receives the message and estimates RTT as $RTT_{measured} = t_{r4} - t_{r2}$

Both peers use an IIR filter for a tight round trip delay estimation (RTT_{est}).

$$RTT_{est} = (1-\alpha)RTT_{measured} + \alpha RTT_{est} \tag{2.1}$$

Discovery procedure require a time period $T_{Discovery}$ that can be calculated as

$$T_{Discovery} = \overline{T_{check}} + timeout_{discovery} \cong timeout_{discovery} \tag{2.2}$$

where $\overline{T_{check}}$ is the average value of T_{check}, the time period used by the P3P peer to trigger synchronous event such as start of a new synchronous procedure, detection of timeout expirations, ...

In equation 2.2 $\overline{T_{check}}$ takes into account of the time period between the event that trigger the discovery procedure and the effective discovery start. Nevertheless $\overline{T_{check}}$ is significantly smaller than the average RTT that is also significantly smaller than $T_{Discovery}$.

Discovery procedure is performed for the first time when Q reads the *.p3p* file that contains the address of the *rendez-vous* node but is also performed when parent node have to relocate the dedicated outgoing resources (*reject procedure*, section 2.5.6.4) and also when Q tries to optimize its position into the multicasting tree (*optimization procedure*, section 2.5.6.6).

Discovery procedure is essentially a client-side procedure: receiving a discovery request does not modify any execution flow in the server-side execution flow graph, figure 2.10. As depicted in figure 2.7 the discovery procedure could be executed both in the basic flow that in the optimization flow.

As explained the main difference between these two invocations of the same procedure is the behavior at the $timeout_{discovery}$ expiration time; if any peers do not reply yet, i.e., no probed peer exist, during the optimization loop the peer starts to unlock its own subtree finishing its optimization attempt but, in the basic loop, the discovery procedure will be repeated.

Moreover, during the execution in the optimization flow, discovery procedure could be stopped in order to perform a sub-tree lock triggered by parent node.

2.5.6.3 Join procedure

Unlike discovery procedure the join one leads to resource allocation and must be reliable in order to avoid resource wasting. Moreover it involves both client side functionalities and server-side ones.

Q node identifies the *parent candidate* node, R, it sends a *join request* to it, figure 2.13, and starts the $timeout_{join}$ countdown. If the $timeout_{join}$ expires without any R reply the request will be repeated.

R receives the request, evaluates the Q *child metric* and always sends a *join reply.* The join reply message can be

accept reply if R can immediately allocate outgoing resources. Moreover R start immediately the $timeout_{join\ reply}$ countdown.

refuse reply if Q child metric values is smaller than the R child node smallest metric one

freeze reply if R can not immediately allocate outgoing resources (no resources available or is performing another join) and the Q child metric are larger than the smallest one of the current R child nodes.

Q becomes a *frozen child* node in the R locale image. If the amount of frozen child node overcomes the maximum allowable child amount, R sends a refuse reply message to the worst frozen child node.

Q receives the R join reply and changes the R state characterization from *parent candidate* node to, respectively,

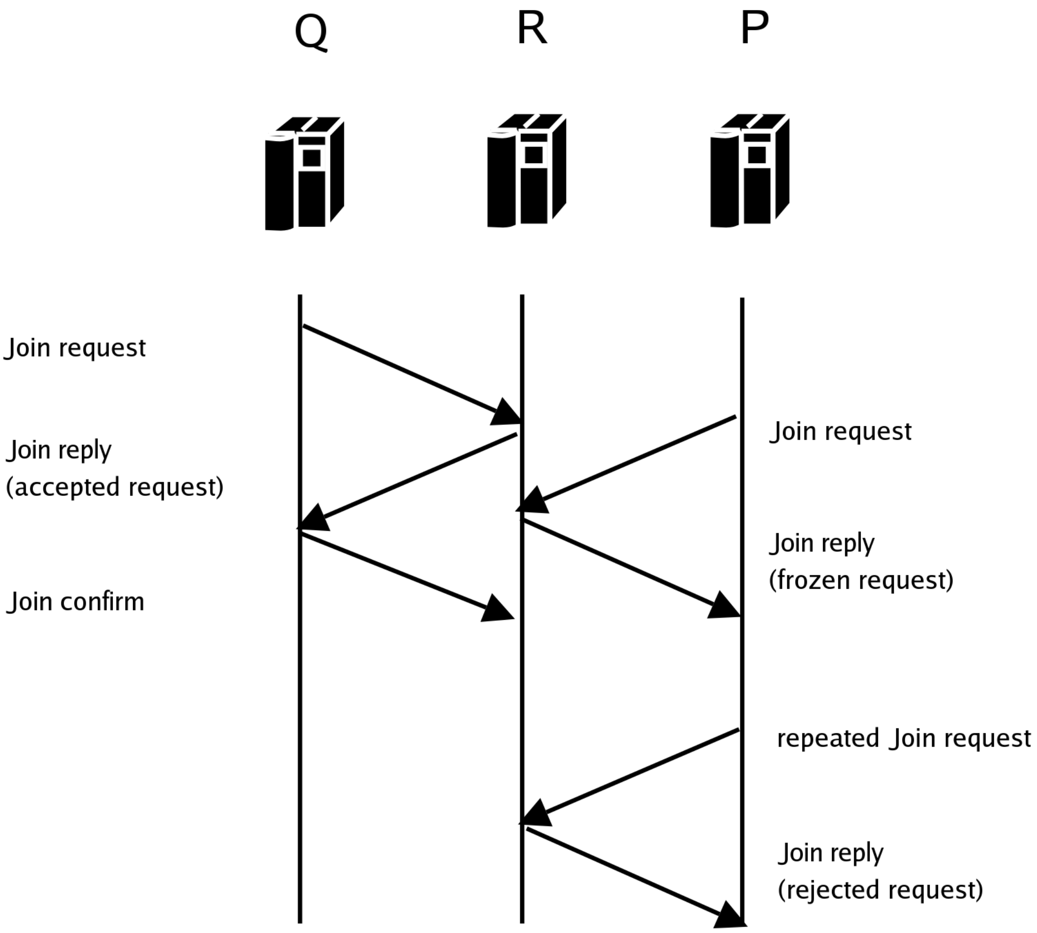


Fig. 2.13: Join phase signaling, *R accept the Q request and freeze the P request until the Join phase with Q is terminated.*

accept reply – **parent *node*** and it sends a *join confirm* message to R. The join confirm message can be used to reset the join procedure (*negative confirm*). For example Q can send a negative confirm message if the R attributes are changed between the discovery procedure and the join one and they become not interestings for Q.

refuse reply – **unknown *node*** therefore Q selects a different *parent candidate* node and repeats the join procedure. If there is no *parent candidate* node Q repeats the discovery procedure

freeze reply – **parent candidate** (the state does not change). Q waits for the $timeout_{waiting}$ expiration and repeats the join procedure to R.

Moreover if the attributes of R will change (the parent metric will decrease) between the discovery procedure and the new join one Q selects another parent candidate node. Eventually, if R sends a second accept reply message, Q will reply with a negative join confirm message.

If the join confirm message is not received before the $timeout_{join\ reply}$ expiration, R sends a new accept reply message and reset the timeout.

At the join confirm reception R allocates the requested resources, i.e., starts to forward the requested content to Q. Obviously Q becomes an R *child* node.

Resources allocation have to be atomic procedure hence each join procedure has to be completed before a new one can begin. Moreover the current join procedure can not be stopped by external messages.

At the join procedure end, if *frozen nodes* exist (brown flow of picture 2.10), R evaluates the larger *child metric* of the current frozen child nodes and, if it is larger than the smallest one of the current R child nodes, it performs a new join procedure with this new joining child peer.

R sends *accept reply* message to the best frozen node and this peer can immediately send a *join confirm* message. Otherwise, if the best frozen child metric value is smaller than the smallest current child node one, R flushes all the frozen node by sending them a *refuse reply.*

In the hypothesis of a negligible packet loss the client side of the join procedure requires a time period T_{join} that can be upper bounded by

$$T_{join} \leq 3 \cdot RTT + (n_{child} - 1) * T_{reject} \cong (n_{child} - 1) * T_{reject} \tag{2.3}$$

The worst situation in order to join to R is when R can not immediately allocates resources and R has got $(n_{child} - 1)$, i.e., the maximum amount of child node minus one, frozen child nodes with larger child metric than Q.

Nevertheless the time required in order to perform the procedure with a peer that can accept immediately the request is

$$T_{join} = RTT \tag{2.4}$$

A more accurate time estimation is calculated on section 2.7.

2.5.6.4 Reject procedure

Reject procedure is executed if a parent node has to relocate outgoing resources in order to accept a joining child node whose child metric is larger than the smaller child node one.

Like the join procedure reject one leads to resource management and hence it should be a reliable procedure.

This is a typical situation: R receives a request from Q; Q child metric is larger than the smaller R child node one, hence S, the worst R child node, has to be replaced by Q. R set the Q state to *frozen child* and set S state as a *rejected child*.

Soft handover policy imposes that the R reallocation resources have to follow a successful S new parent node search.

In order to increase packet loss resiliency R periodically sends a *reject request* message until S will reply to it.

At the *reject request* message reception, S set R state to *bad parent* node and begins a parent search.

Like join procedure the reject one is performed both on server side (Reject macro states of figure 2.9) that in client side (figure 2.6). In the server side flows reject procedure is atomic and can not be stopped by any external messages. On the client-side receiving a reject request message produces a transition toward basic flow (the new parent search will be executed with the higher priority).

At the end of parent node search S sends a *reject reply* message to R and set R state to *unknown node*. At the message reception R reallocates the resources dedicated to S and completes the join procedure with its *best* frozen child (the best one is not necessarily the first come node).

Moreover the *reject reply* message can be used to notify to the current parent node the success of an optimization procedure (deallocate loop in figure 2.9). Node that migrates to another parent node and join to it has to notify to the previous parent to deallocate its outgoing resources. This message should be sent using a reject reply one.

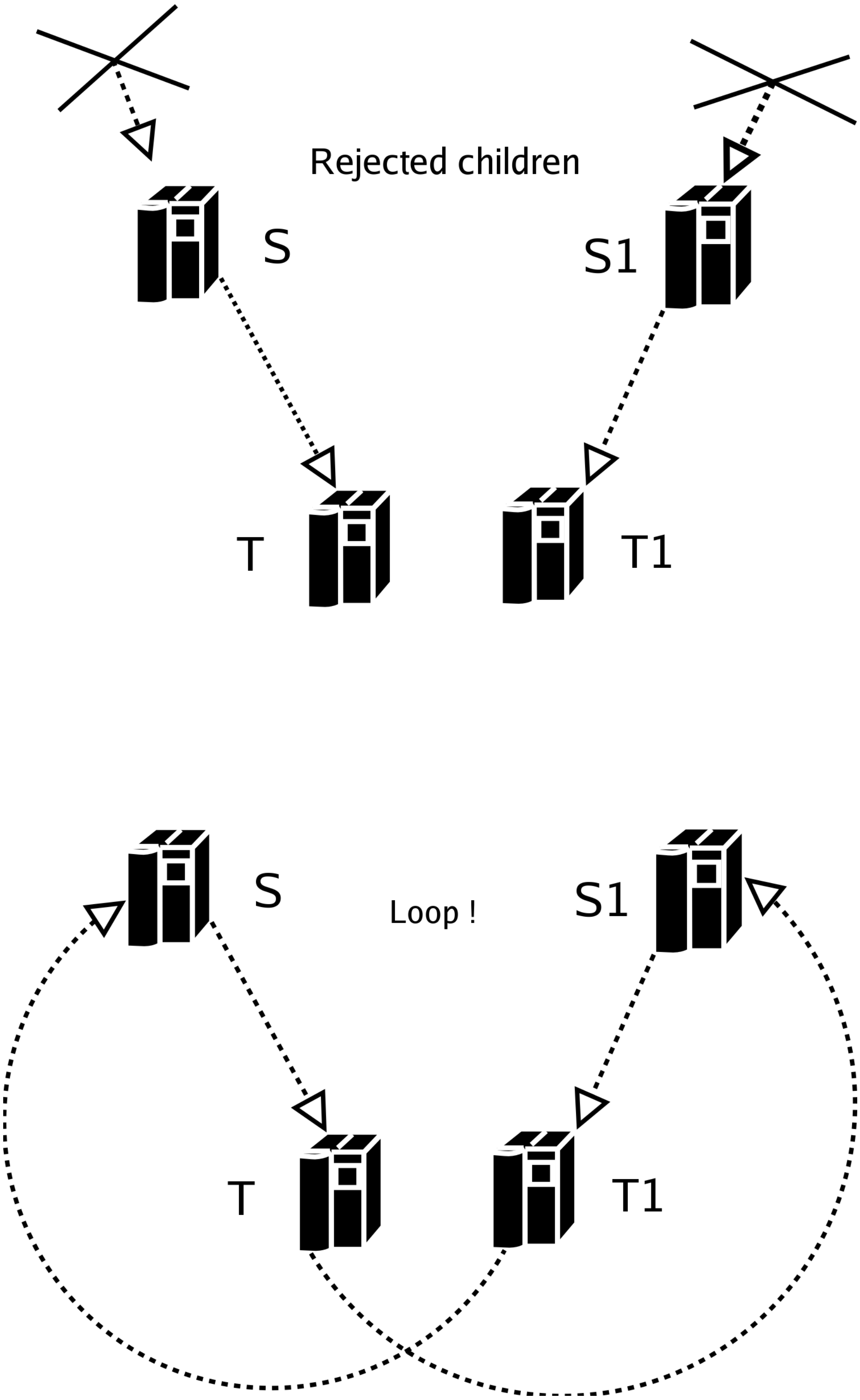


Fig. 2.14: Loop situation

In the hypothesis of a negligible packet loss the client side execution of the reject procedure requires a time period T_{reject} that can estimated as

$$T_{reject} = T_{lock} + T_{discovery} + T_{join} + RTT \tag{2.5}$$

In order to complete the reply procedure a node has to find a new parent and it has to complete successfully a new join. This time period involves the execution of other procedures (Lick, Discovery, Join). A more accurate estimation of the required time is calculated on section 2.7.

2.5.6.5 Lock - Unlock procedure

In order to avoid creation of peer loop sequences each discovery and join procedures don't start until all the peer in the subtree are inhibited to accept join request. In figure 2.14 is depicted a possible loop generation due to unlocke subtree and contemporary node migration.

Each peer knows only the peers that communicate directly with it (neighbor nodes) and can not be able to previews that other peers will try to complete successfully a join procedure with nodes in the own sub-tree.

Lock and unlock procedures can be driven by the peer itself or by an ancestor node. The loop avoidance execution flow involves both the client side and the server side peer execution flows.

Lock and unlock procedure should be reliable and lock or unlock requests are sent to respectively unlocked and locked child until all the child nodes reach the requested state.

A peer sends a reply to its parent node only if whose child nodes are all locked or all unlocked. In picture 2.15 this behavior is depicted.

Request and reply messages are propagated across subtree with this paradigm until the requests reach the leaf nodes and the replies come back to the peer that has started the procedure.

In figure 2.15, in the hypothesis that RTT are the same between each couple of peer and that each peer immediately replies or immediately forwards the request to its own child nodes, the process can be completed in a RTT amount related to the tree level of the $P1$ node subtree.

In the hypothesis of a negligible packet loss the lock (or unlock) procedure requires a time period T_{lock} that can estimated as

$$T_{lock}^{(l)} = RTT \cdot l \tag{2.6}$$

where l is the amount of level of the distribution sub-tree whose root is the peer that begins the lock procedure.

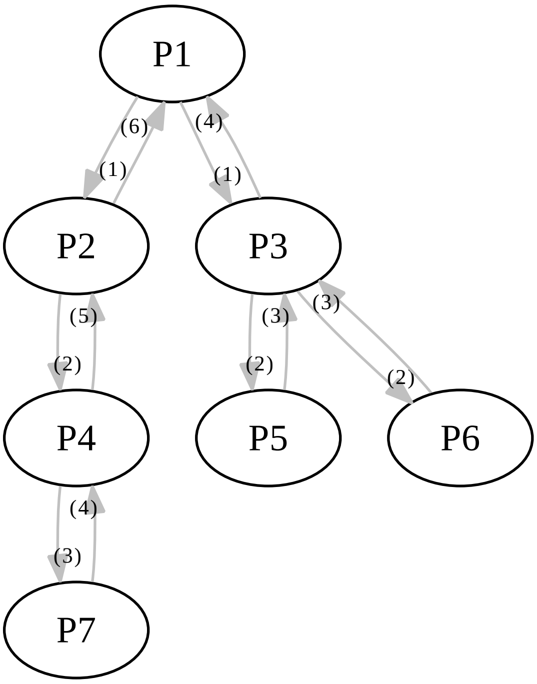


Fig. 2.15: lock or unlock paradigm

If a node receives a *lock request* during the execution of the join procedure it will reply only after the conclusion of the current join. This behavior leads to takes into account of an additive delay value in the range $(0, T_{join})$, but also it allows to stop the *lock request* propagation into the tree section, i.e., a joining node subtree is always locked.

However the following upper bound can be written

$$T^{(l)}_{lock} \leq RTT \cdot (l + 1) \tag{2.7}$$

2.5.6.6 Optimizing procedure

The optimizing procedure is performed in order to find abetter position into the P3P tree structure. The procedure starts at the $timeout_{sleeping}$ expiration that will be reset at the reception of each message that is not used in the automatic recovery procedure. $timeout_{sleeping}$

- is initialized at each unlocking procedure at a value that decreases as long tree position level increases.

- duplicates after each unsuccessful optimization attempt.

These strategies allows to realize a stable mechanism that promotes migrations of peer far from the root node.

The optimization procedure is essentially a client-side routine that triggers a sequence of lock-discovery-join procedure executed with lower priority (green flow on figure 2.6)

2.5.6.7 Recovery procedure

The recovery procedure is periodically executed by each peer in order to evaluate the neighbor node conditions. Each peer begins a three way handshake with its own parent node and at the end of this handshake, if the parent node is running, it will estimate the RTT between itself and the parent node. Thanks to the third way message each parent node can also estimates the RTT between itself and its child nodes.

If parent node does not reply for a time period longer than $timeout_{dead}$ it will be considered a failed node and a new parent search is started (basic flow on figure 2.6). Moreover if any messages are not received from a child peer for a $timeout_{dead}$ period the child node will be considered down and the reserved resources will be free, see figure 2.11 on section 2.5.5.

2.5.7 Tree Management Protocol: initializing and updating node references

In order to evaluate child metric and parent metric each node has to publish its own attributes. Some characteristics are decided autonomously by the peer but other ones are related to the position into each distribution tree structure or to the amount of neighbor nodes.

Each peer decides autonomously

- the total outgoing bandwidth and how this can be partitioned across the distribution trees in order to promote a distribution tree. Each node decides how many child nodes can be provided in each tree.
- which distribution tree will be joined or not.

Nevertheless each node decides all the other attributes following the tree management protocol rules. In detail each node updates

- the amount of its current child nodes in each single tree

- its own level, i.e., the amount of peer between the node itself and the tree root in each tree.
- the sequence of ancestor node in each tree
- the round trip time between itself and the neighbor nodes.

All this second set of attributes have to be continuously updated.

In each outgoing P3P messages each node attaches all its own current attributes and all the nodes that receive these messages can learn other nodes characteristics.

If an executed procedure leads to resources management, the attributes will change accordingly. This change are related only to the peer that performs the procedure and has not to be propagated.

Instead, if the executed procedure leads to a migration into the tree structure, the attribute change has to be propagated into the peer sub-tree. Locking the subtree ensures that wrong attributes could not be sent away to the subtree and, during the unlocking procedure, the new attributes can be updated thanks to the unlocking request messages that are sent by parent nodes.

2.6 Software implementation

P3P multimedia stream management and signalation are written in *C++*. Standard *C++* does not support multi threading execution and hence non standard libraries have been used: *Boost C++ libraries*, [13].

Boost libraries are intended to be widely useful and usable across a broad spectrum of applications. Ten Boost libraries are already included in the C++ Standards Committee's Library Technical Report (TR1) and will be in the new C++0x Standard now being finalized. C++0x will also include several more Boost libraries in addition to those from TR1. More Boost libraries are proposed for TR2.

P3P is composed by 2 UDP Socket, *RTPSocket* and *ControlSocket* and the following resources shared by 5 different threads:

TargetList Is the collection of the current child nodes grouped by distribution tree

IncomingControlQueue Is the incoming queue of P3P signalation messages

OutgoingControlQueue Is the outgoing queue of RTCP messages P3P signalation messages

P3Pcore Is the real core of the P3P software: it contains all the peer states, all the neighbors information and all the peer attributes

The task of each thread is the following

RTP Management This thread manages the incoming RTP flows and also sends the received packets to all the other peers that are identified as child node in the received packet distribution tree. It reads the RTP streams from *RTPSocket*, neighbor addresses from *TargetList* and then it forwards the packets using *RTPSocket*.

RTCP and Control Receiver This thread reads the P3P incoming messages and the incoming RTCP messages from *ControlSocket*. Moreover it pushes P3P messages in the *IncomingControlQueue* and RTCP packets in the *OutgoingControlQueue*.

RTCP and Control Sender This thread reads P3P messages or RTCP messages from *OutgoingControlQueue* and it pushes them in the *ControlSocket* reading the new destination addresses from *TargetList*.

Asynchronous Logic This thread reads incoming P3P packets from *IncomingControlQueue* and it manages them by following the P3P management protocol rules. It modifies the Peer state, *P3Pcore*, and it updates *TargetList*

Synchronous Logic This thread periodically checks timeout expiration or scheduled procedure execution. As the previous thread it modifies the Peer state, *P3Pcore*, and it updates *TargetList*

In figure 2.16 the interactions between sockets, threads, and shared resources are depicted.

2.6.1 P3P packet header

P3P messages are built on top of UDP header. In figure 2.17 the P3P message header is depicted

C *(mandatory), 2 bits* Code field distinguishes between RTP/RTCP messages and P3P messages. P3P Code is $0x0$.

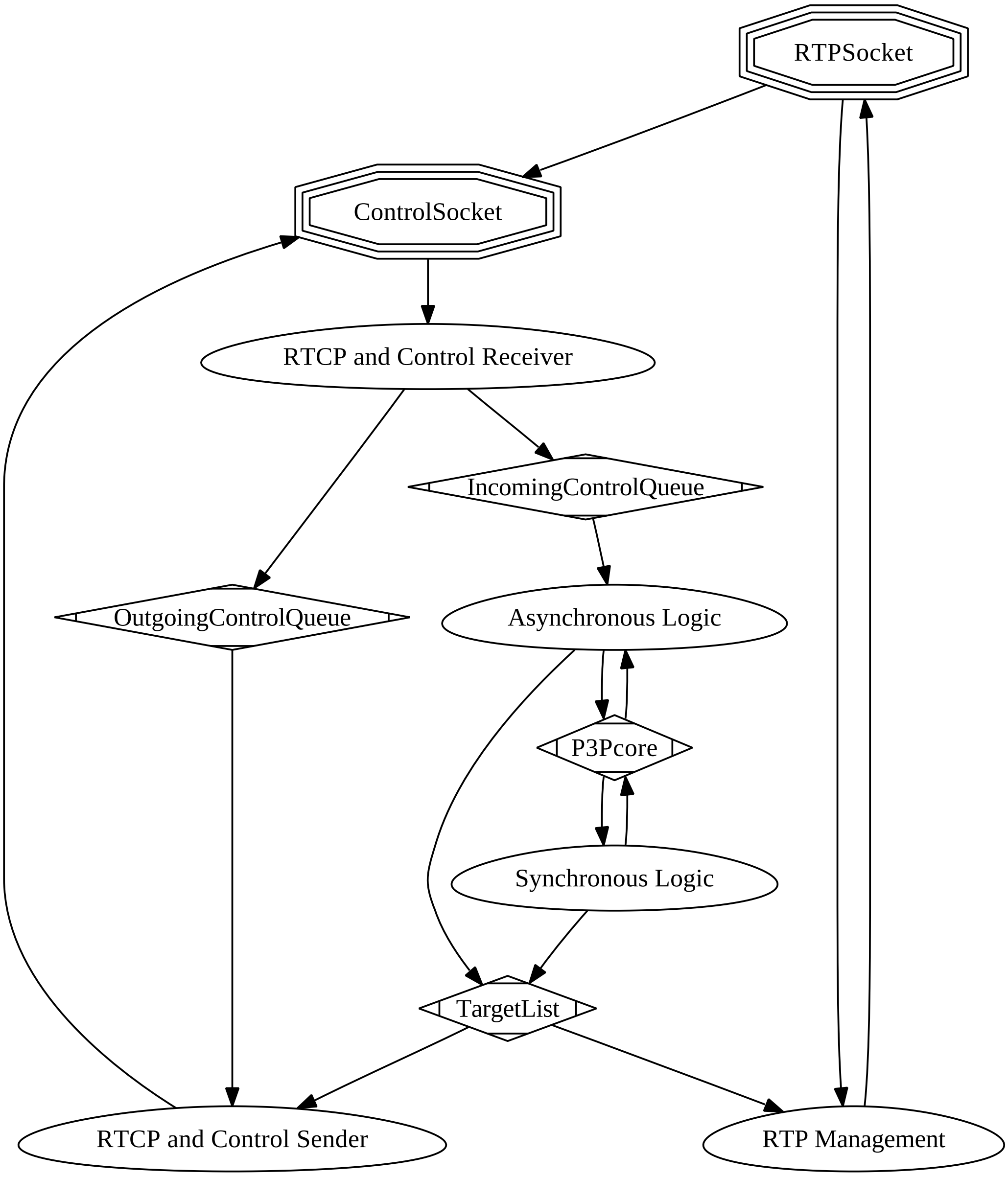


Fig. 2.16: P3P software entities

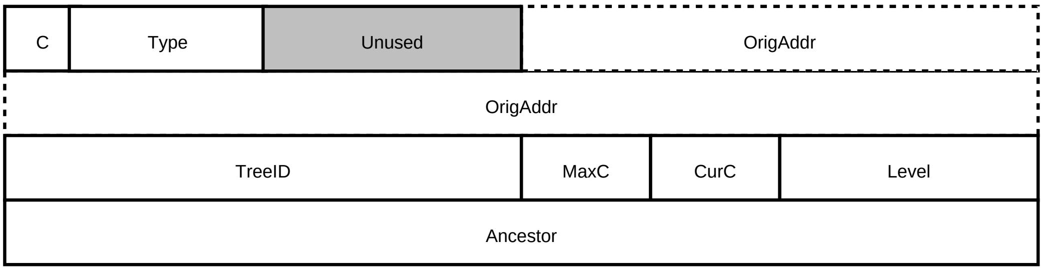


Fig. 2.17: P3P message header

Type *(mandatory), 6 bits* Type field permits to identify the type of P3P message (Discovery Request, Join Accept reply, . . .).

TreeId *(mandatory), 16 bits* Identifier of the received P3P message distribution tree.

MaxC *(mandatory), 4 bits* The maximum allowable child nodes amount in the identified distribution tree.

CurC *(mandatory), 4 bits* The current child nodes amount in the identified distribution tree

Level *(mandatory), 8 bits* The current nodes amount between the peer and the root tree node.

Ancestor *(mandatory), 32 bits* Product of the nodes identifier of the peers between the node itself and the root tree node, ancestor node. Each node is identified by a prime number. This field allows to identify if a node is or not an ancestor node

OrigAddr *(optional), 48 bits* Address of the peer that begins the discovery procedure. It is used in order to send a discovery reply directly to the node that originates the procedure. This field is not mandatory: it makes sense only if field *Type* indicates a *Discovery request* message.

2.6.2 RTP and RTCP packet management

P3P has to tag each RTP or RTCP packet as belonging to a single distribution tree identified by a 16 bits identifier. RTP packet can be extended with the aid of RTP header extension, [14], depicted in figure 2.18. If the X bit in the RTP header is set to one, a variable-length header extension is appended to the RTP header, following the CSRC list if present. The header extension contains a 16-bit length field that counts the number of 32-bit words in the

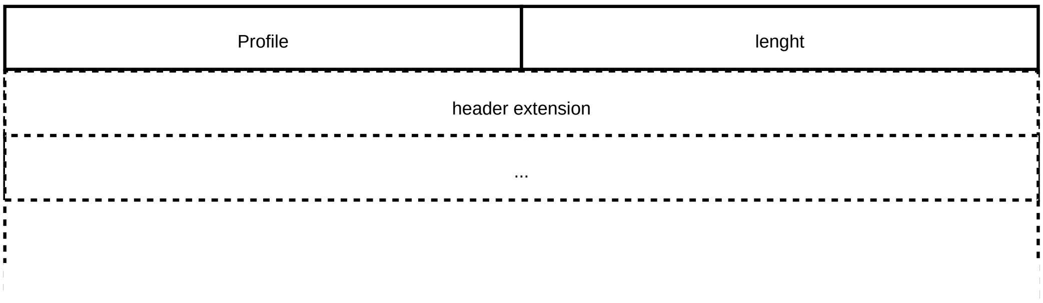


Fig. 2.18: RTP header extension

extension, excluding the four-octet extension header (therefore zero is a valid length). Only a single extension may be appended to the RTP data header.

Therefore P3P can tag each RTP packet setting X bit in the RTP header and adding a 0 word RTP extension in which the field *profile* contains the distribution tree identifier and the field length is set to 0.

2.7 Connection time analysis

In section 2.5.6 the time required in order to accomplish each analyzed procedure is estimated in equations 2.2, 2.4, 2.5, 2.7. Nevertheless some procedure can require the execution of another procedures and also requires interactions with other nodes that are probably performing other procedures.

Therefore a tight overall time estimation can not be reduced to the sum of each single execution time over the sequence of required procedures (at least because a deterministic sequence does not exist).

The time period required to connect to a parent peer is pivotal in all the following situation:

- at the first connection to each P3P distribution tree (P3P client side)
- after a parent node failure detection (P3P client-side)
- after the first *reject request* message reception (P3P client side)
- after an acceptable join request if the available resources are exhausted (P3P server side)

Nevertheless connection time involves also the optimization procedure duration and each ancestor node driven lock phase.

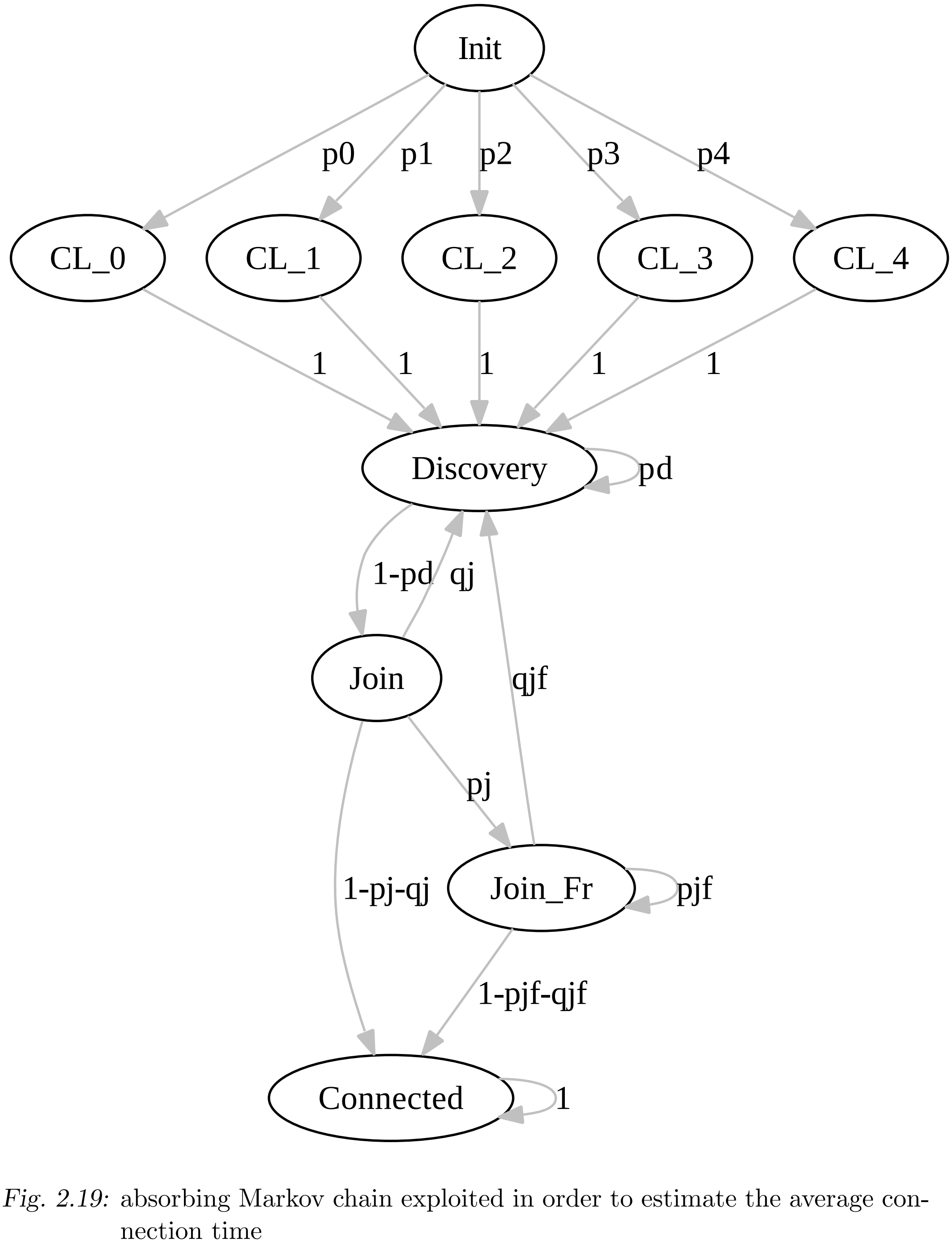


Fig. 2.19: absorbing Markov chain exploited in order to estimate the average connection time

In order to evaluate the overall time required to connect to a new parent peer the Markov chain representation of figure 2.19 can be exploited.
Connection is the only absorbent state ($p_{connected,connected} = 1$, $p_{connected,i} = 0\ \forall i \neq connected$) of the chain, i.e., the chain is an absorbing one.

The process execution time of each one of the N states of this Markov chain can be fixed or upper bounded by specific values T_i. Therefore defining T as the $\{X_n\}$ Markov chain random absorption time

$$T = min\left\{n \geq 0; X_n = connected\right\} \tag{2.8}$$

the average value of $T^{(e)}$, the process execution time, can be calculated as

$$\begin{aligned} E\left[T^{(e)}\right] &= E\left[\sum_{n=0}^{T}\sum_{i=0}^{N}\delta_i(n)\cdot T_i|X_0 = Initial\right] \\ &= \sum_{i=0}^{N} E\left[\sum_{n=0}^{T}\delta_i(n)|X_0 = Initial\right]\cdot T_i \\ &= \sum_{i=0}^{N} W_{Initial,\ i}\cdot T_i \end{aligned} \tag{2.9}$$

in which $W_{i,k}$ determines the mean number of visit to state k prior to absorption starting from state i

At first, in all the listed situations a peer has to lock its own subtree; $T_{lock}^{(l)}$, equation 2.7, depends at least from the amount of level, l, of the peer subtree and the amount of levels depends also on the amount of peers.
In the following the results involve only 10 peers interaction processes and therefore a maximum amount of 4 level subtree is assumed.

Markov chain state CL_v models the situations in which the locked sub-tree has got v different node levels.

In the examined process *discovery phase* can lead both to another *discovery phase* and also to a *join phase*. Nevertheless the time required to complete each discovery procedure, $T_{discovery}$, is fixed to $timeout_{discovery}$ as showed in equation 2.2.

Join phase can lead to discovery one, to complete the connection to a new parent peer or also to repeat the join procedure. In order to take into account of *join freeze* reply and the following $timeout_{waiting}$ expiration the Markov chain *join Fr* state is created.
The time duration of the *normal* join phase is about RTT, the time required to receive and evaluate the accept or the refuse reply, but if the join reply is a frozen one, the peer will wait the $timeout_{waiting}$ expiration and after that it tries to complete the join procedure. Therefore $T_{Join} = RTT$ and T_{JoinFr} is equal to $timeout_{waiting}$

Unlock phase can be forgotten in this analysis because from the client side point of view the new connection ends at the join reply message reception.

The figure 2.19 Markov chain transition probability matrix is

$$\underline{\underline{P}}_{N\times N} = \begin{pmatrix} 0 & p_0 & p_1 & p_2 & p_3 & p_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & p_d & 1-p_d & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & q_j & 0 & p_j & 1-p_j-q_j \\ 0 & 0 & 0 & 0 & 0 & 0 & q_{jf} & 0 & p_{jf} & 1-p_{jf}-q_{jf} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.10}$$

that can be written as

$$\underline{\underline{P}}_{N\times N} = \begin{pmatrix} \underline{\underline{Q}}_{N-1\times N-1} & \underline{\underline{R}}_{N-1\times 1} \\ \underline{\underline{0}}_{1\times N-1} & \underline{\underline{I}}_{1\times 1} \end{pmatrix}$$

$\underline{\underline{W}} = (W_{i,k})$, the matrix that contains the mean number of visit to state k prior to absorption starting from state i, accordingly to the first step analysis theory, can be calculated as

$$\underline{\underline{W}} = \left[\underline{\underline{I}}_{N-1\times N-1} - \underline{\underline{Q}}\right]^{-1}$$

In detail, fixing *Initial* as the starting state, the mean amount of visit to state k can be calculated as

$$\begin{aligned} W_{Initial,CL_0} &= p_0 \\ W_{Initial,CL_1} &= p_1 \\ W_{Initial,CL_2} &= p_2 \\ W_{Initial,CL_3} &= p_3 \\ W_{Initial,CL_4} &= p_4 \\ W_{Initial,discovery} &= \frac{1-p_{jf}}{(1-p_d)\big((1-q_j)(1-p_{jf})-p_j*q_{jf}\big)} \\ W_{Initial,join} &= \frac{1-p_{jf}}{(1-q_j)(1-p_{jf})-p_j*q_{jf}} \\ W_{Initial,joinfr} &= \frac{p_j}{(1-q_j)(1-p_{jf})-p_j*q_{jf}} \end{aligned} \tag{2.11}$$

Obviously $W_{Initial,CL_x} < 1$ and depends only to probability p_x. Moreover the probabilities of having x amount of level in the subtree does not affect the mean number of visit of *Discovery*, *Join* and *Join Fr* state.

In addition $W_{Initial,join}$ and $W_{Initial,joinfr}$ do not depend to p_d

Previous equations can be written as

$$\begin{array}{lcl} W_{Initial,discovery} & = & \frac{W_{Initial,join}}{1-p_d} \\ W_{Initial,join} & = & \frac{1}{1-q_j-\frac{p_j*q_{jf}}{1-p_{jf}}} \\ W_{Initial,joinfr} & = & \frac{1}{\frac{(1-q_j)(1-p_{jf})}{p_j}-q_{jf}} \end{array} \tag{2.12}$$

and in order to simplify the model the transition probabilities of *join* and *join fr* Markov chain state can be be fixed to the same values.

$$\begin{array}{lcl} W_{Initial,discovery} & = & \frac{W_{Initial,join}}{1-p_d} \\ W_{Initial,join} & = & \frac{1-p_j}{1-q_j-p_j} \\ W_{Initial,joinfr} & = & \frac{p_j}{1-q_j-p_j} \end{array} \tag{2.13}$$

that means that the average number of visits to state $Join$ prior to absorption is the ration between the probability of repeating the discovery procedure and the probability of complete the connection process. Moreover the average number of visits to state $Join$ prior to absorption is the ration between the probability of repeating the join procedure and the probability of completing the connection process.

Therefore, from equation 2.9,

$$\begin{array}{lcl} E\left[T^{(e)}\right] & = & \sum_{i=0}^{N} W_{Initial,\ i} \cdot T_i \\ & = & \sum_{k=0}^{4} p_k \cdot T_{lock}^{(k)} + \frac{1-p_j}{(1-q_j-p_j)\cdot(1-p_d)} \cdot T_{discovery} + \\ & & \frac{1-p_j}{1-q_j-p_j} \cdot T_{join} + \frac{p_j}{1-q_j-p_j} \cdot T_{joinfr} \\ & < & \sum_{k=0}^{4} p_k \cdot (k+1) \cdot RTT + \frac{\frac{timeout_{discovery}}{1-p_d} + RTT\cdot(1-p_j) + (timeout_{waiting})\cdot p_j}{1-q_j-p_j} \end{array} \tag{2.14}$$

2.8 Evaluating performances

In the following a 10 peer interaction is considered with-in 2 different distribution trees. Both the distribution tree are originated by the same node, *Root Peer*.

Each peer can allocate resources for the amount of nodes of table 2.1 In pictures 2.20 2.21 2.22 the behavior of nodes that constitute the first distribution tree is showed. At first *Root* directly allocates its own resources and then nodes start to join the first level child nodes (third step).

Child metric promotes nodes with larger outgoing resources and then node *P1* has to search a new parent. Between the fourth and the fifth step *P1*

Peer	Child node amount (first tree)	Child node amount (second tree)
Root	3	3
P1	1	1
P2	2	2
P3	3	3
P4	2	2
P5	2	2
P6	1	1
P7	1	1
P8	1	1
P9	1	1

Tab. 2.1: Peer outgoing resources configuration

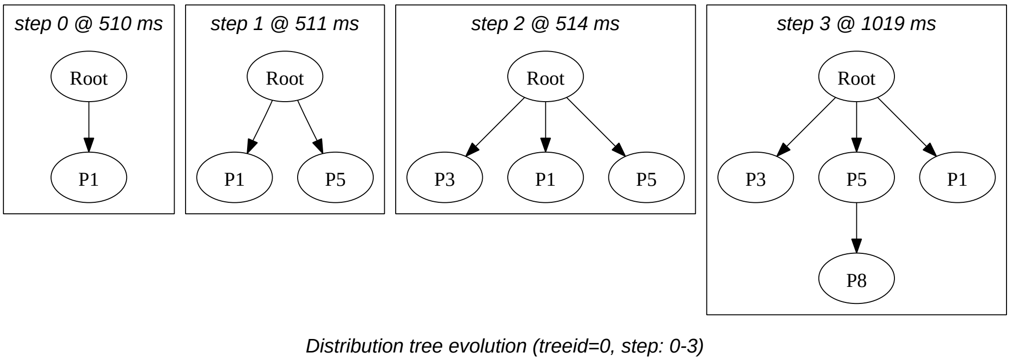


Fig. 2.20: Evolution of the first distribution tree

send a reject reply to the *Root* node that in the fifth step accepts node *P2*. In the ninth step *P1* completes the join procedure and becomes a child node of *P3*.

In the tenth step *P9* completes an optimization procedure and migrates from *P3* to *P2*: the experimented RTT between the couple *(P9,P2)* is smaller than the one between *(P3,P2)* and parent metric promotes nodes with lower RTT (if parent candidate nodes are at the same level).

Figures 2.23, 2.24, 2.25 show the second distribution tree evolution. The distribution tree grows with a sequence of step slightly different from the first one but at the last step the distribution trees are identical.

In a hand this effect leads to the conclusion that the obtained structure is the best one related to the child and parent metric bounds and to the underlying network conditions. On the other hand identical distribution tree frustrates

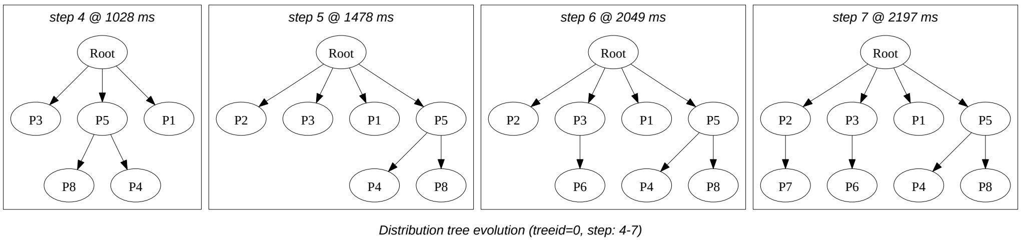

step 4 @ 1028 ms
step 5 @ 1478 ms
step 6 @ 2049 ms
step 7 @ 2197 ms

Fig. 2.21: Evolution of the first distribution tree

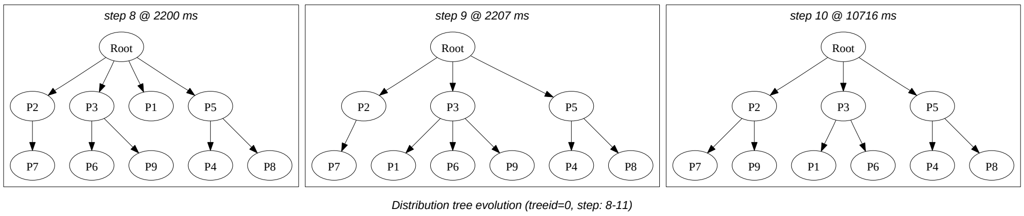

step 8 @ 2200 ms
step 9 @ 2207 ms
step 10 @ 10716 ms

Fig. 2.22: Evolution of the first distribution tree

the path redundancy and the resilience advantage that is assured by multiple description coding.

2.8.1 Bushy tree strategies

As explained in section 2.5.2 the distribution trees have to be as short as possible and evolve following the rules decided by child and parent metric formulation.

Nevertheless building distribution trees requires that first come nodes have to migrate in order to relocate resources to node with greater child met-

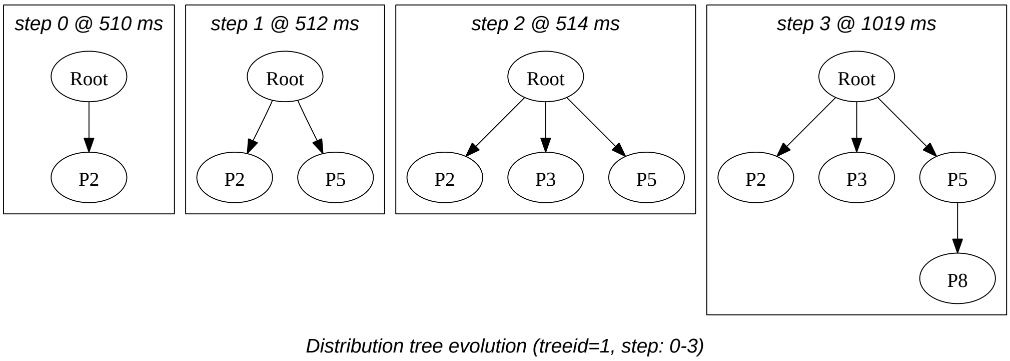
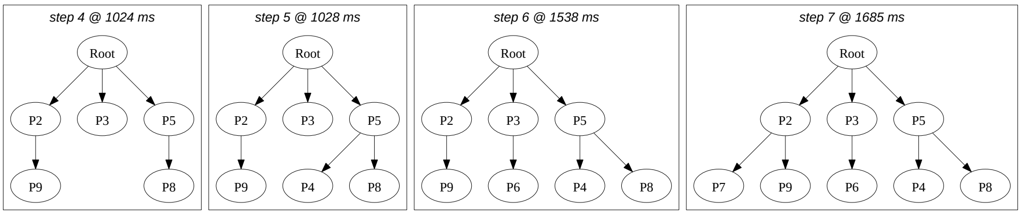

step 0 @ 510 ms
step 1 @ 512 ms
step 2 @ 514 ms
step 3 @ 1019 ms

Fig. 2.23: Evolution of the second distribution tree


step 4 @ 1024 ms

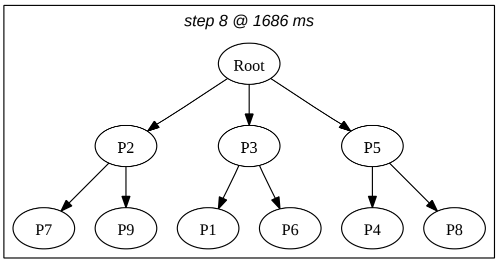
step 5 @ 1028 ms

step 6 @ 1538 ms

step 7 @ 1685 ms

Distribution tree evolution (treeid=1, step: 4-7)

Fig. 2.24: Evolution of the second distribution tree


step 8 @ 1686 ms

Distribution tree evolution (treeid=1, step: 8-11)

Fig. 2.25: Evolution of the second distribution tree

ric. Therefore some nodes that were initially connected at some level in the distribution tree could migrate to larger level and wait for an optimization phase (driven by themself or by ancestor nodes) in order to obtain a better position.

Therefore in each distribution tree realization some nodes may live in a level that is greater than the optimal one. In figure 2.26 is depicted the cumulative amount of time in which each node is connected to a position greater than the optimal one related to the simulation duration. The optimal level is calculated using the peer outgoing resources in table 2.1 and is set to 2.

Moreover performances of figure 2.26 is measured in a really harsh environment in which all the join requests arrive within a few milliseconds. In picture 2.27 a more real environment is investigated: each peer begins the discovery procedure after 0.5 seconds from the previous peer start. ed in a really harsh environment in which each join request arrives within a few milliseconds. In picture 2.27 a more real environment is investigated: each peer begins the discovery procedure after 0.5 seconds from the previous peer start.

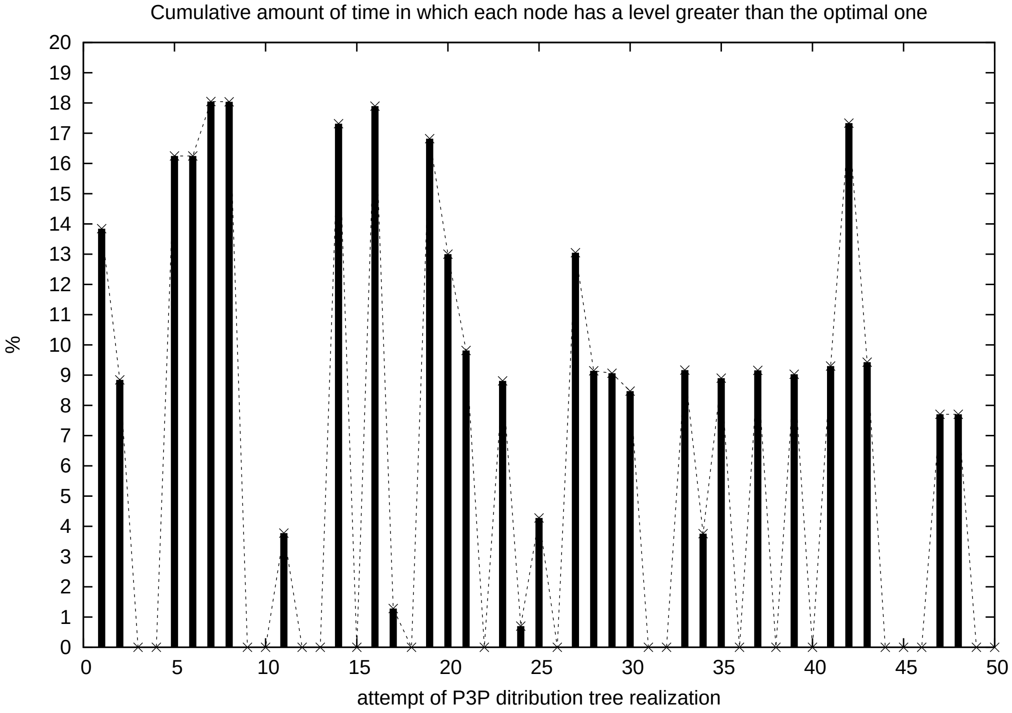


Fig. 2.26: Ratio between the cumulative amount of time in which each node is connected to the distribution tree in a level greater than the optimal one and the time duration of the measure, harsh environment

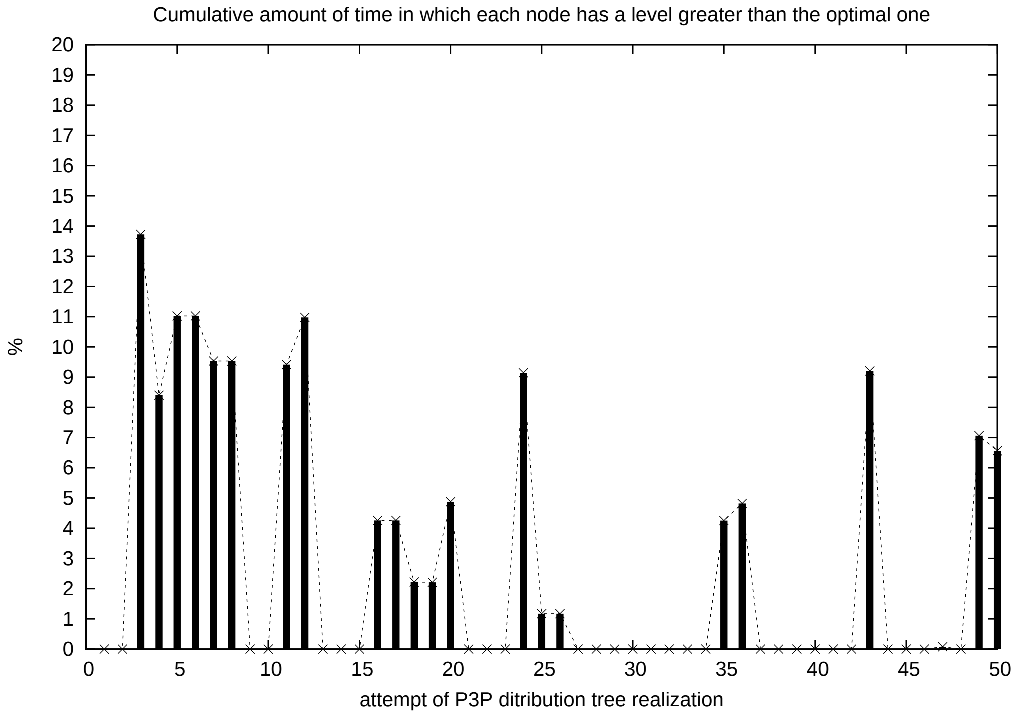


Fig. 2.27: Ratio between the cumulative amount of time in which each node is connected to the distribution tree in a level greater than the optimal one and the time duration of the measure, quiet environment

Peer	Child node amount (first tree)	Child node amount (second tree)
Root	2	2
P1	1	1
P2	2	2
P3	2	2
P4	2	2
P5	2	2
P6	2	2
P7	1	1
P8	1	1
P9	1	1

Tab. 2.2: Peer outgoing resources configuration

2.8.2 Distribution tree orthogonality

As depicted in pictures 2.22 and 2.25 different distribution trees are built adopting the same rules (fixed by parent metric and child metric) and even the step are different the final tree realization is the same.

In order to increase the resilience of the overall system against peer failure in parent metric and also in child metric the sequence of ancestor node in each distribution tree is compared with the ancestor sequence in the other tree in order to minimize the common ancestor node amount.

The effect of this metric review on both the distribution tree is depicted in figures 2.28, 2.29, 2.30, 2.31 and in figures 2.32, 2.33, 2.34, 2.35 in which the evolution of two coupled distribution tree is showed.

In order to increase the possibility of realize more orthogonal distribution tree the resource allocation of table 2.2 is used. Each peer continues to declare the same outgoing resources in both the distribution tree but the optimal tree maximum level is increased to 3. Moreover an entire class of nodes (**P2,P3,P4,P5,P6**) could obtain one of the two available slots in the first level of nodes.

In order to evaluate the effect of the previous metrics review the cumulative amount of common ancestor node over all the peer is investigated. In figure 2.36 the improved metric, the one in which the information related to the ancestor sequence is considered, outperforms the previous one. Indeed without considering ancestor sequence both the distribution tree become the tree required by child metric and parent metric formulation and each node has got exactly the same ancestor nodes in both the distribution tree. The introduction of ancestor sequence allows to compare and amend accordingly

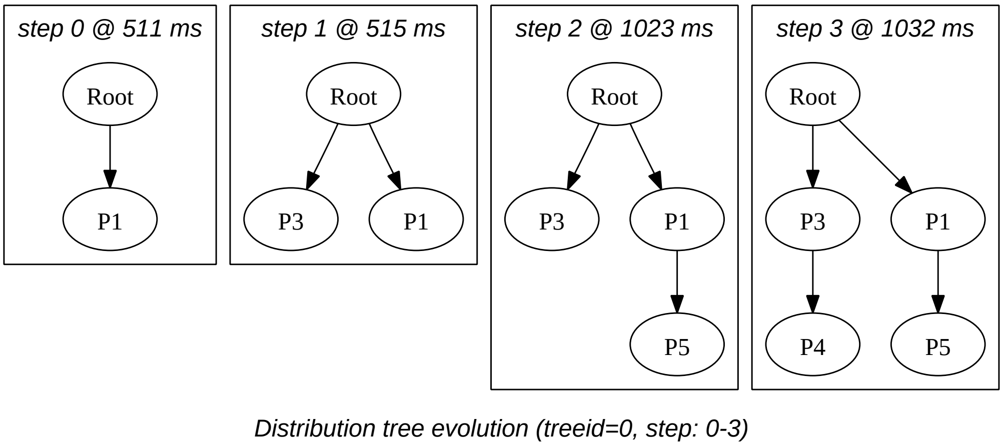


Fig. 2.28: First distribution tree evolution

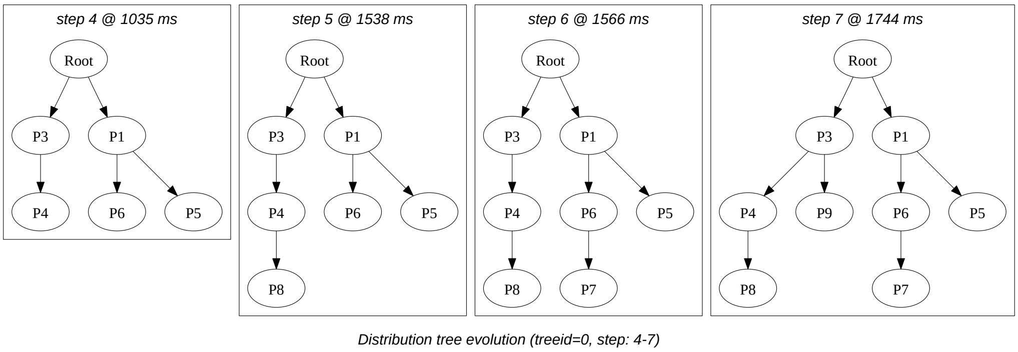


Fig. 2.29: First distribution tree evolution

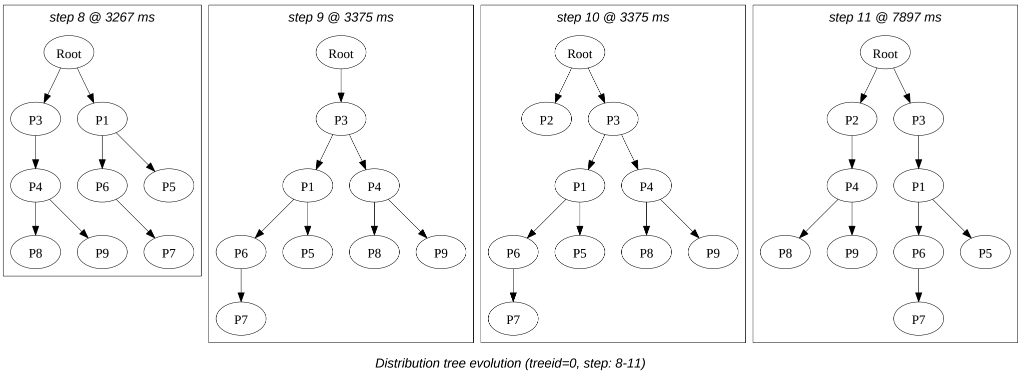


Fig. 2.30: First distribution tree evolution

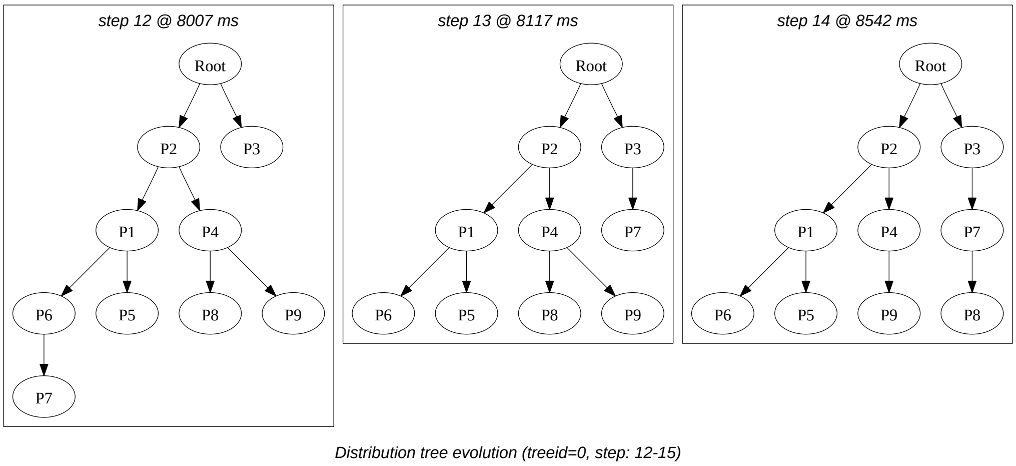


Fig. 2.31: First distribution tree evolution

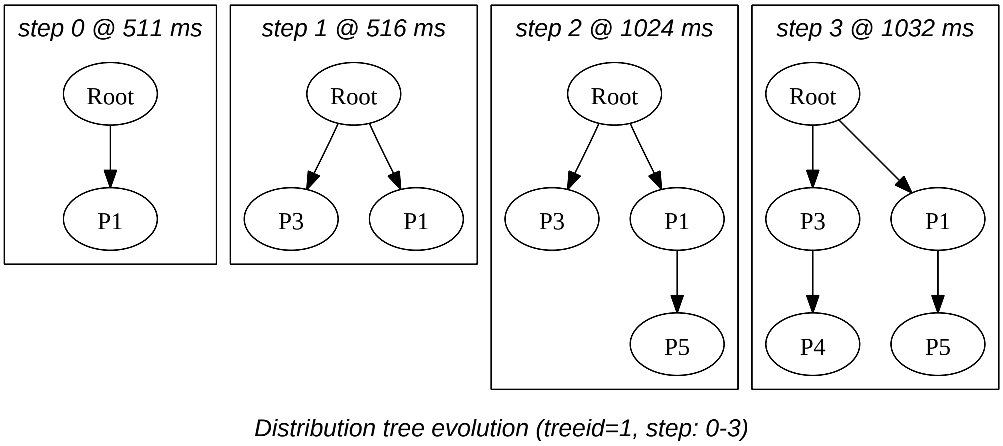


Fig. 2.32: Second distribution tree evolution

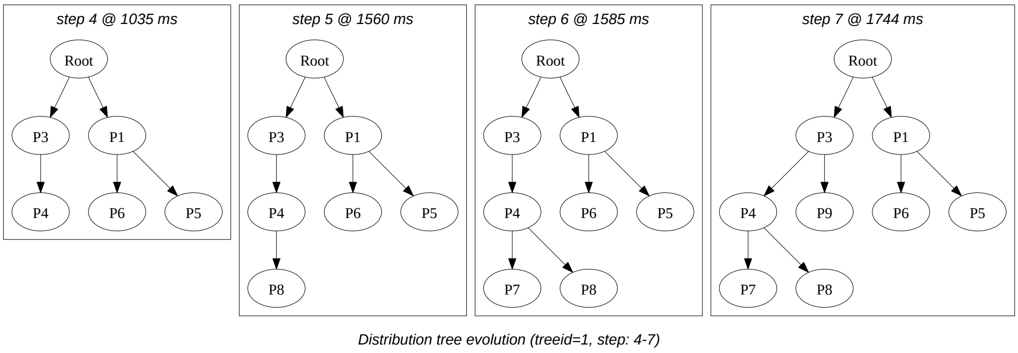


Fig. 2.33: Second distribution tree evolution

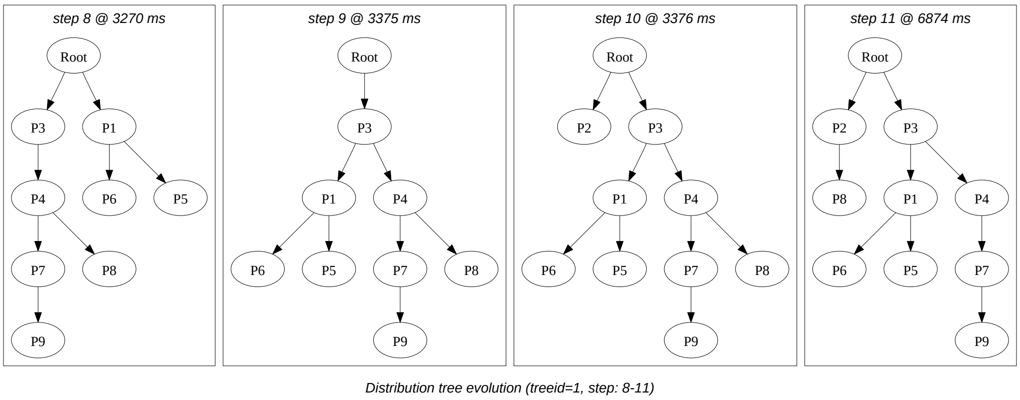


Fig. 2.34: Second distribution tree evolution

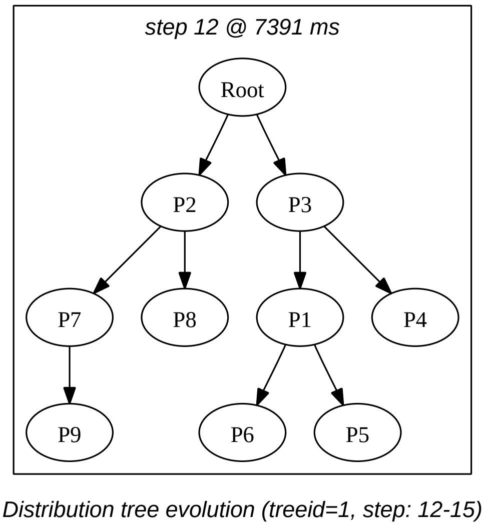


Fig. 2.35: Second distribution tree evolution

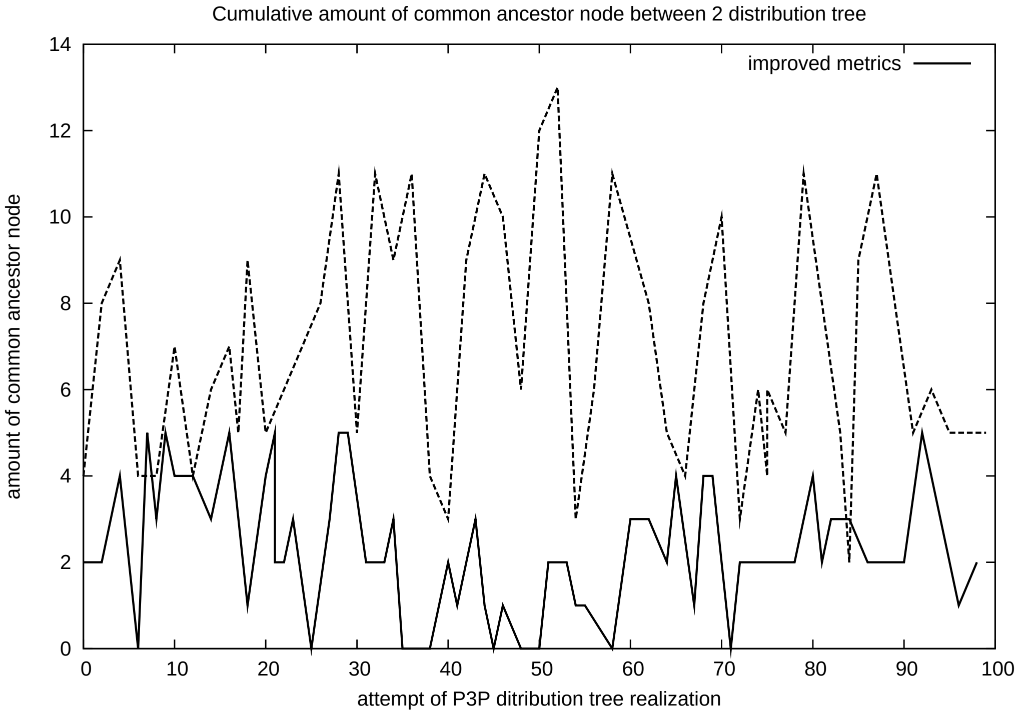


Fig. 2.36: Cumulative amount of common ancestor node for each peer between the distribution tree

the evolution of the distribution tree.

Another strategy that could be used in order to obtain a stronger tree orthogonality is differentiate the maximum declared amount of child nodes in each distribution tree. Nevertheless this strategy affects also the content distribution fairness and should be further investigated.

In figure 2.37 and 2.38 the delay penalty, the delay introduced by the P3P structure, is showed. The test environment is a full gigabit ethernet local area network.

The introduced delay is of the same magnitude than the optimal one and is strictly related to the position of each peer into the distribution tree.

2.8.3 Resilience against node failure

As explained P3P uses a multiple description code, and every description are delivered to all the peer through an independent multicasting tree. With this approach a peer failure does not completely compromise the service delivery in its own subtree.

In order to show hoe P3P react to detected failure the following configuration

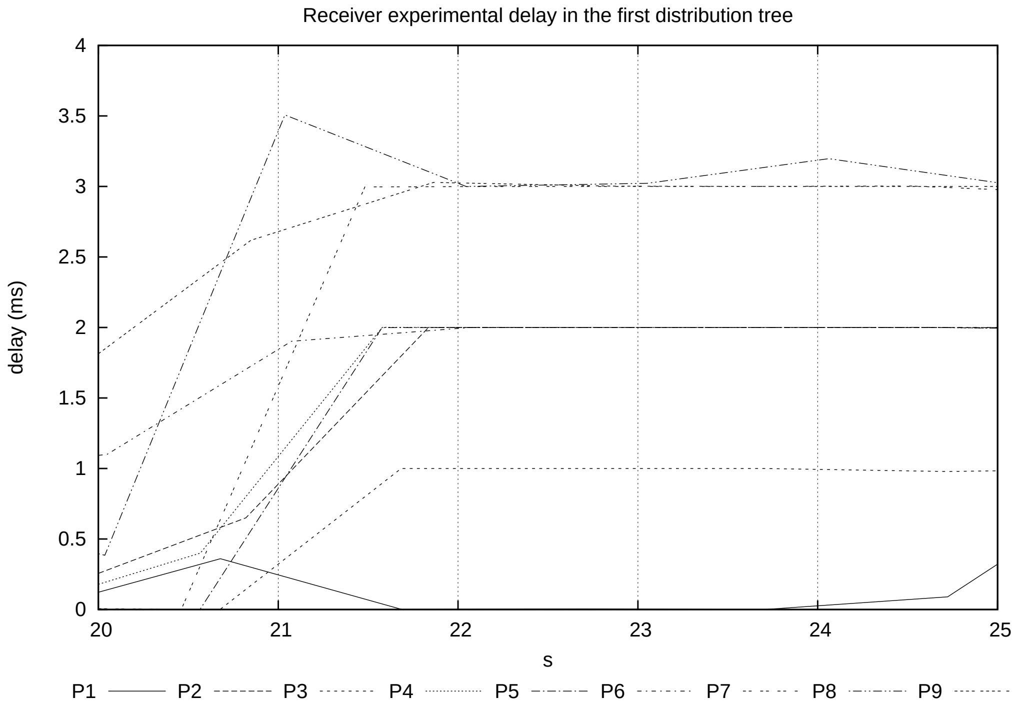


Fig. 2.37: Relative delay penalty

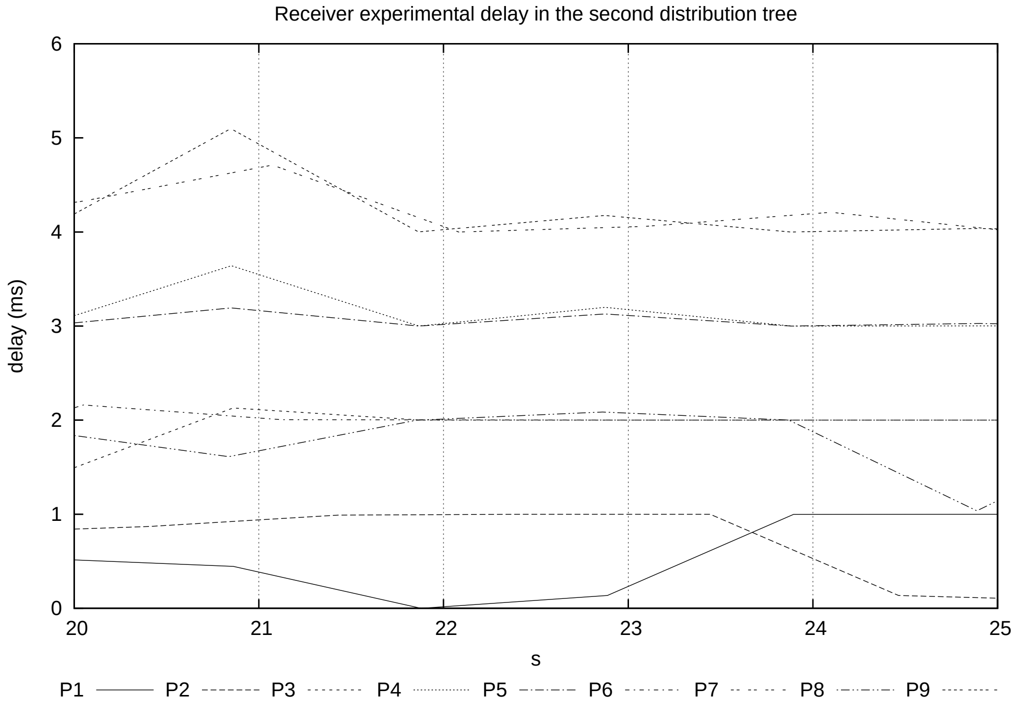


Fig. 2.38: Relative Delay Penalty

Peer	Child node amount (first tree)	Child node amount (second tree)
Root	2	2
P1	1	1
P2	2	2
P3	3	3
P4	3	3
P5	2	2
P6	2	2
P7	1	1
P8	1	1
P9	1	1

Tab. 2.3: Peer outgoing resources configuration

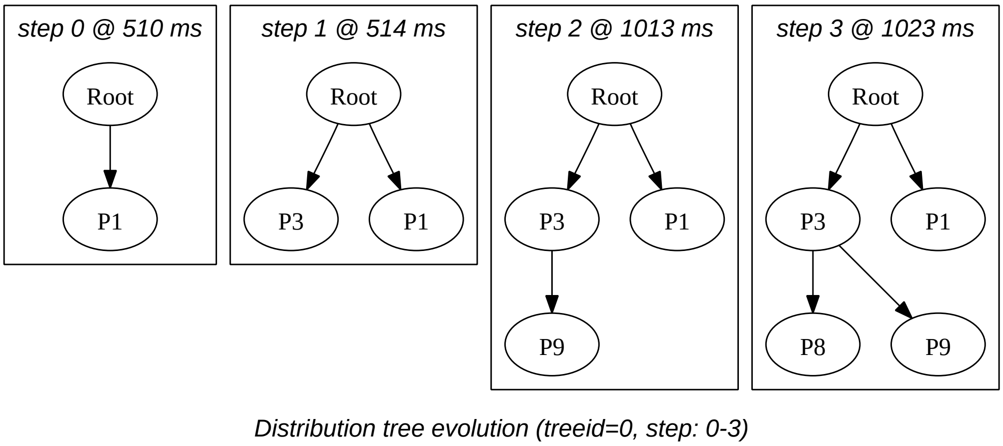


Fig. 2.39: Tree behavior during a peer failure event

(table 2.3) is adopted and one of the peers with the large amount of child is killed 10 seconds after the begin of P3P session the tree evolution is shown in figures 2.39, 2.40, 2.41, 2.42 .

Initially P3 and P1 become child nodes of the root, After that P4, whose child metric is larger than the P1 one, substitutes P1 and all the P1 subtree migrate in the P3 subtree. After that P3 fails and all the node in its subtree have to migrate. P1 immediately moves in the P4 subtree and than it becomes a root node child. In figure 2.43 all the P1 movement are depicted together with the moment of the Reject request message reception and with the moment of parent failure detection.

In figure 2.44 is depicted the effect of parent failure on P1 packet loss. Packet

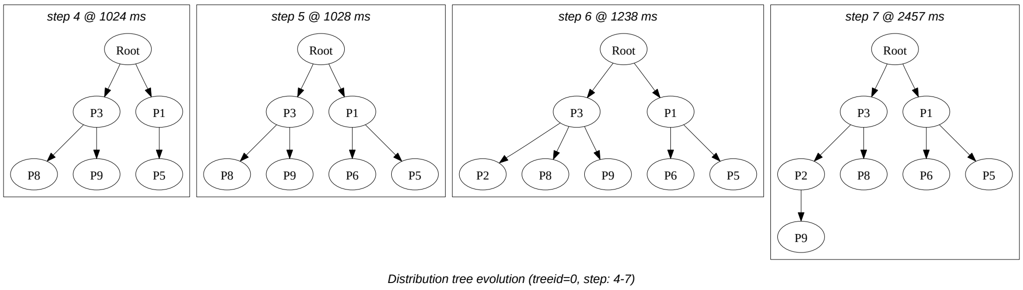


Fig. 2.40: Tree behavior during a peer failure event

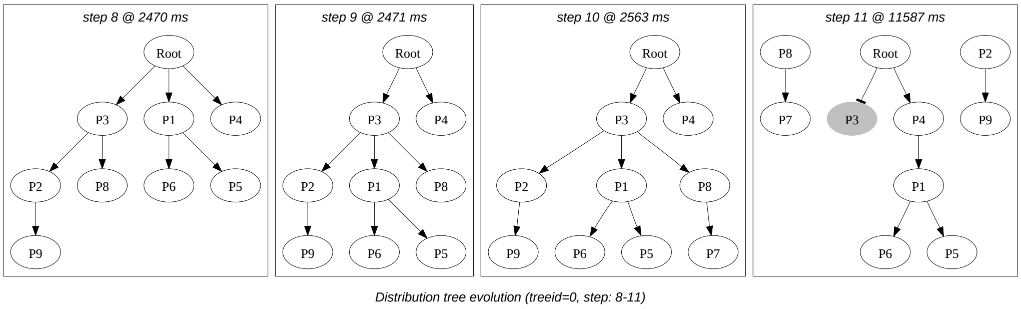


Fig. 2.41: Tree behavior during a peer failure event

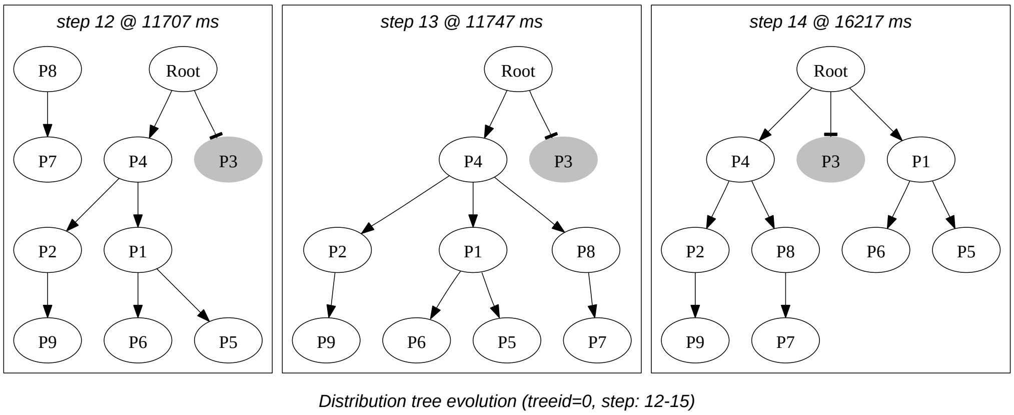


Fig. 2.42: Tree behavior during a peer failure event

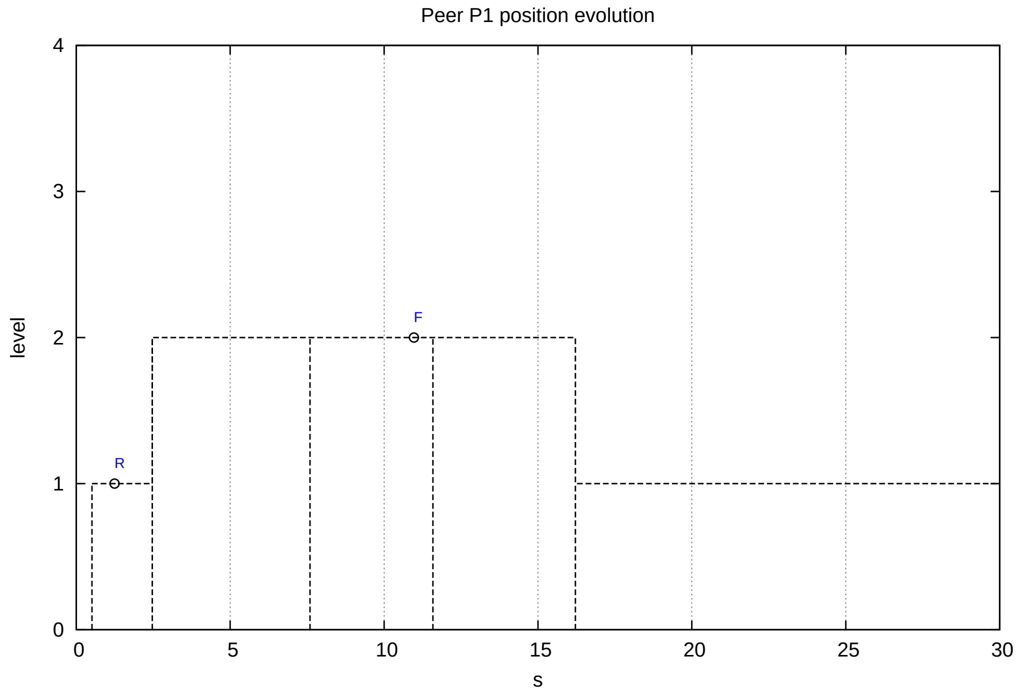


Fig. 2.43: P1 movement into the first distribution tree during a peer failure event

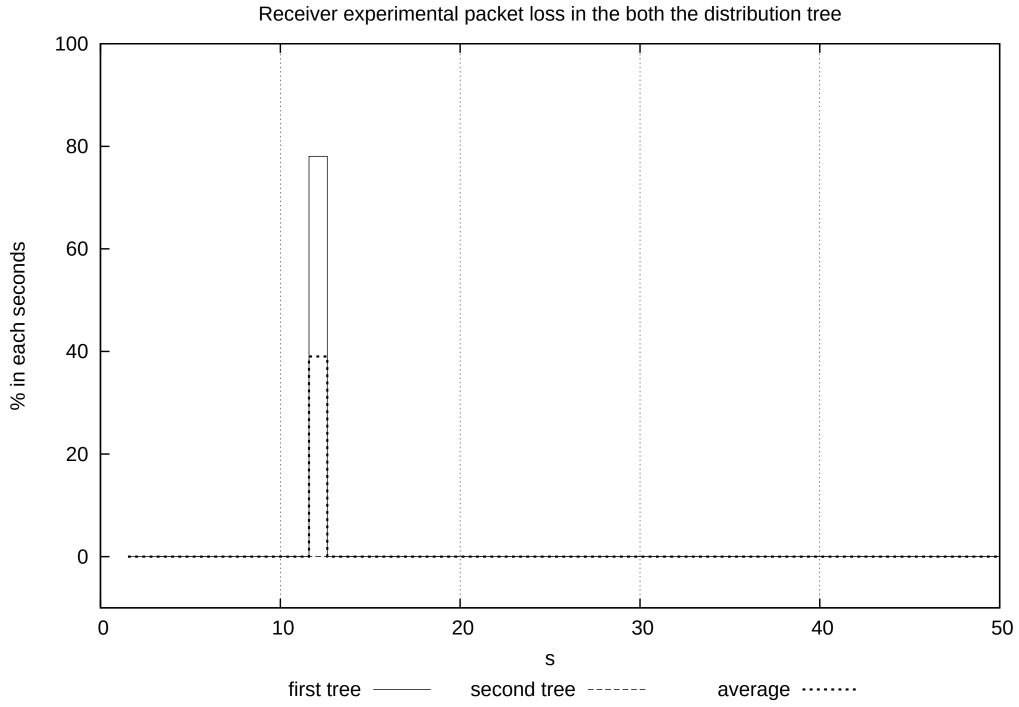


Fig. 2.44: P1 perceived packet loss on both the distribution trees and on average

	ms
$timeout_{discovery}$	500
$timeout_{waiting}$	100

Tab. 2.4: Peer outgoing resources configuration

sub-tree level	p_j	q_j
2	0.4	0.02
1	0.3	0.2
0	0.25	05

Tab. 2.5: Markov model probabilities variation

loss measure is performed each second but the peer failure affect the received stream only in the second in which it happens and not in the following ones.

Moreover parent failure does not affect P1 in the second distribution tree therefore, on the average, a user experiments only a quality degradation and not a service interruption.

2.8.4 Reconnection time

In figure 2.43 P1 receives a reject message and after a period that are essentially the time required to complete a connection to another peer in the same distribution tree, it migrates.

In the same figure P1, after detecting the P3 failure, searches another parent and than migrates.

In section 2.7 equation 2.7 find an upper bound of the reconnection time.

In the following discussion the P3P parameter in table 2.4 is used. Moreover experimental measures yield to neglect p_d and highlight correlation between the amount of subtree level and the couple (p_j, q_j). In table 2.5 is show how (p_j, q_j) changes varying the amount of levels of the peer subtree

Intuitively p_j decreases and q_j increases because the probability of being accepted as child node increases if peer has got a large child node metric, i.e., lots of outgoing resources, and large child node metrics leads to obtain a low level into the distribution trees. Therefore nodes with large child metric if can not be immediately accepted are frozen, but nodes with small child metric are often refused.

In figure 2.45 is depicted an collection of node reconnection delay values for node that are root ones of a two level subtree. The estimated value is very

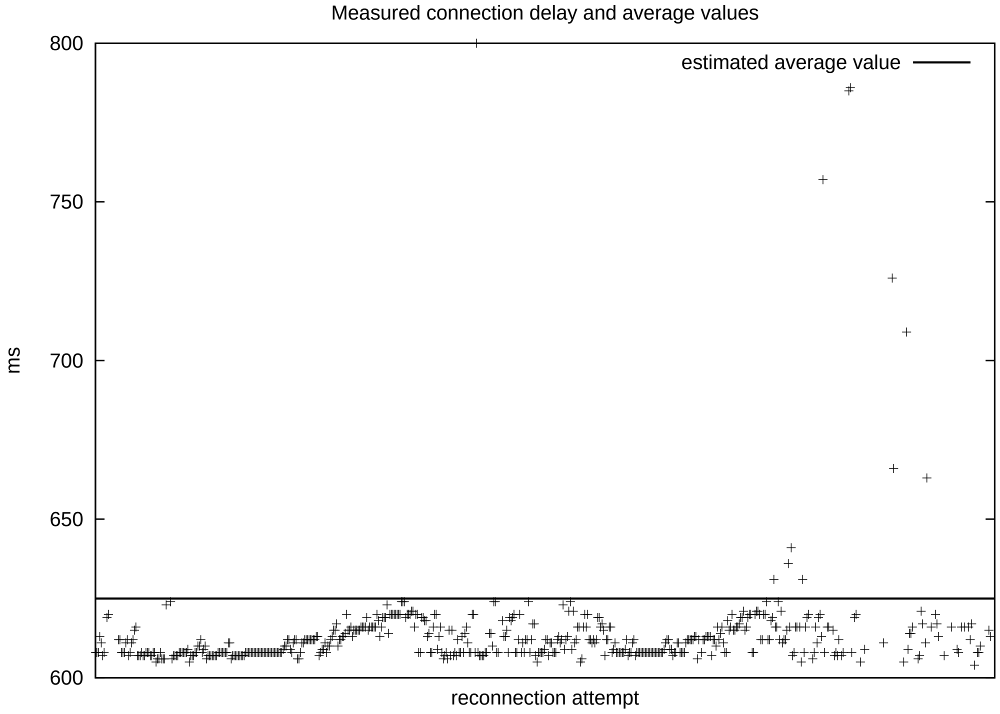


Fig. 2.45: Delay values of (re)connection attempt of 2-level subtree root node

close to the largest part of measured value.

2.8.5 Scalability of signalation traffic

In figure 2.46 the amount of signalation outgoing message for each peer in a 10 peer interaction is depicted. The declared outgoing resources of each peer is the one declared in table 2.2.

The amount of outgoing messages increases linearly with a slope that is related to the amount of child nodes. Indeed the larger part of the outgoing message amount is composed by the alive messages of section 2.5.6.7. If figure 2.47 is shown how message amount can be subdivided by typologies and in picture 2.48 the same data are depicted but *Alive* messages are omitted.

In picture 2.49 the signalation message amount of the overall P3P structure measured in the first 50 second of session is depicted. In order to compare the result the overall value is divided by the amount of peer that participate to the session. The measured values do not significantly increase by increasing the amount of peers that participate to the sessions.

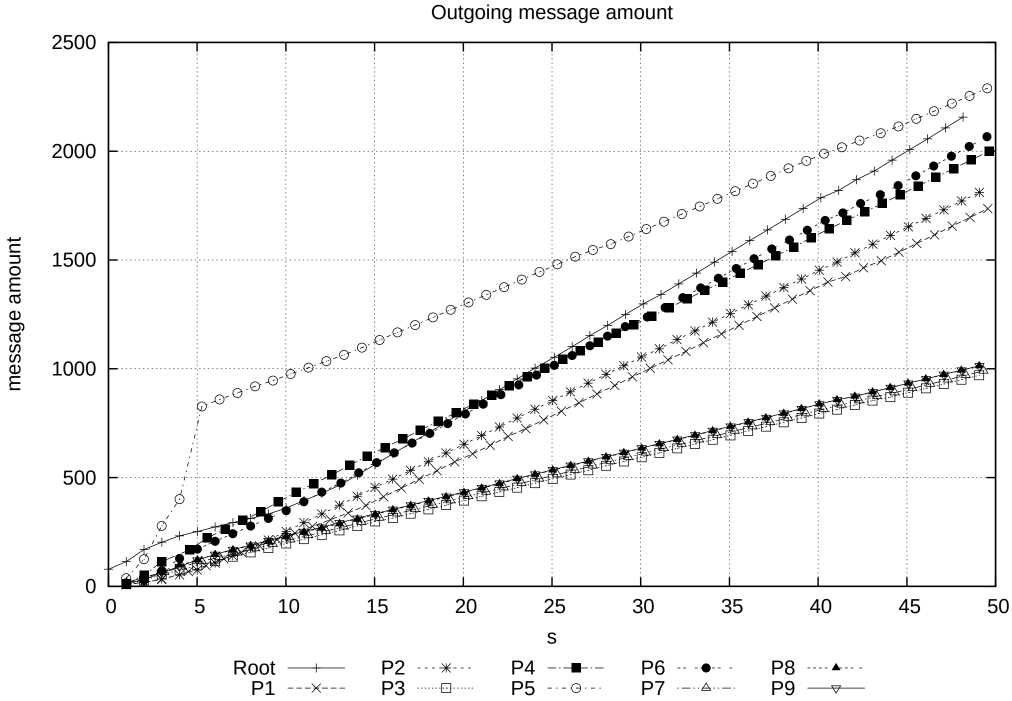


Fig. 2.46: Amount of signalation message divided by peer

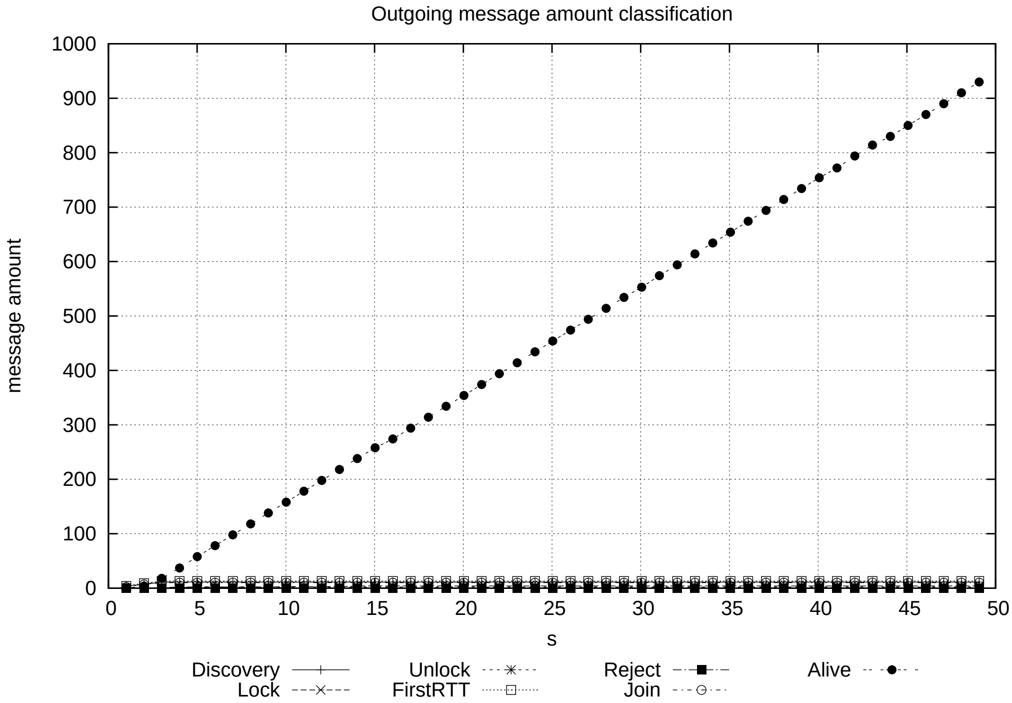


Fig. 2.47: Message amount subdivided by typology and produced by peer P3

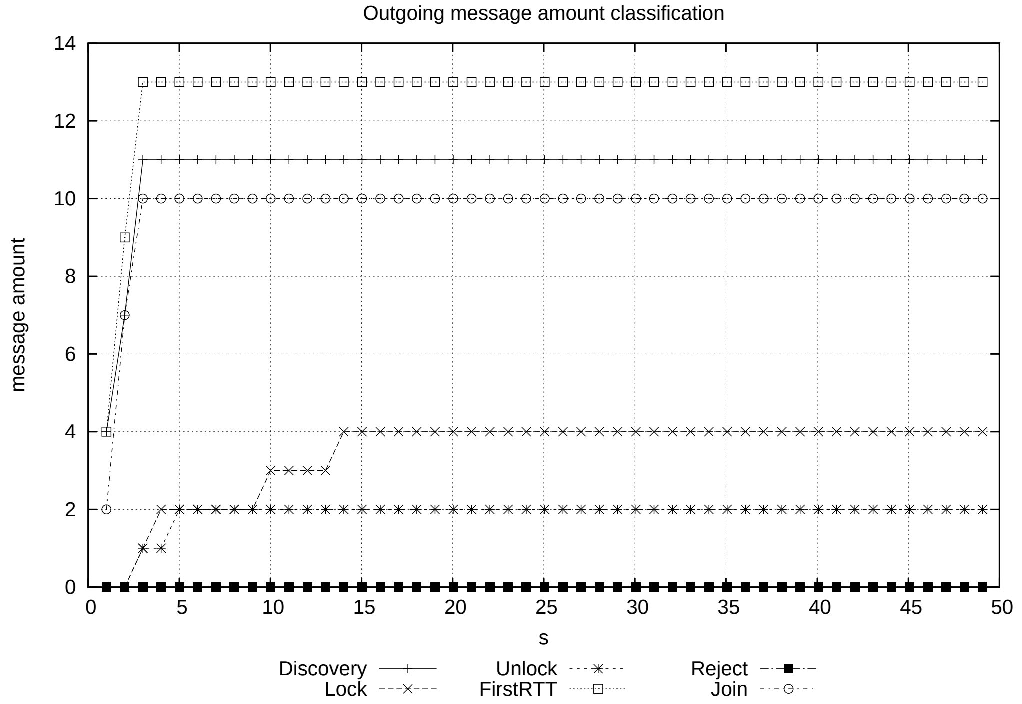


Fig. 2.48: Message amount subdivided by typology and produced by peer P3

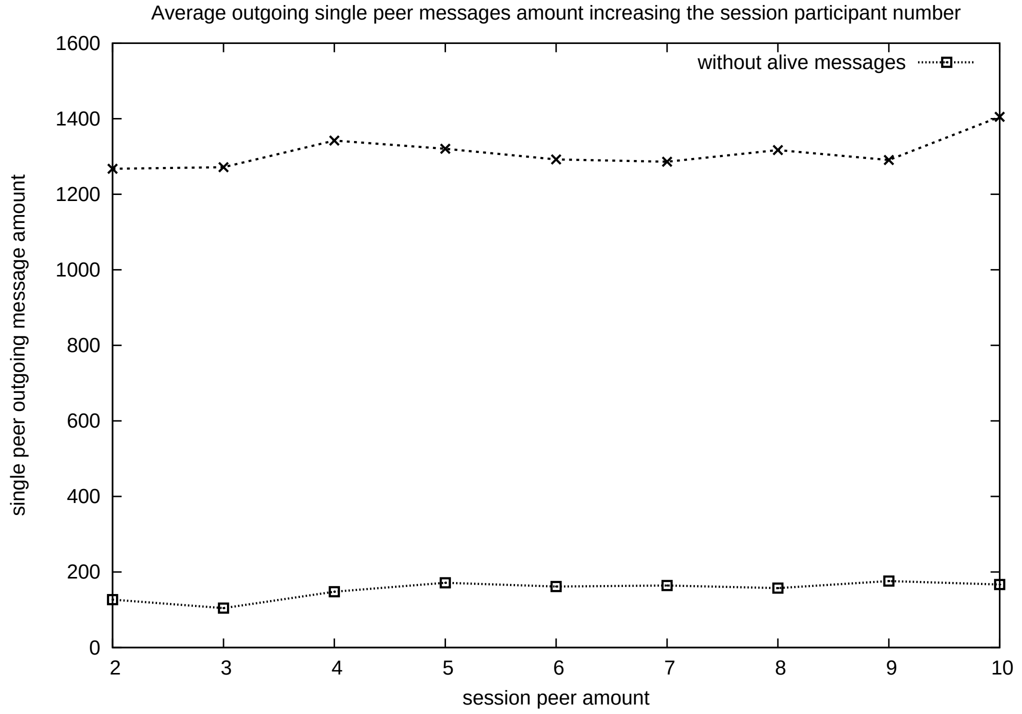


Fig. 2.49: Signalation message amount in the first 50 s of P3P session varying the number of participant

2.9 Conclusions

In this chapter a completely distributed overlay multicast platform is proposed and analyzed.

The aim of this overlay platform is to deliver real time data to a set of peers and allow to a group of participants to join together in a single and unforeseen session without a dedicated infrastructure and without resource wasting.

Discovery procedure permits to the participant to know each other without a rendez-vous server i.e., there is the only need of publish or send by mail a small *.p3p* that contains the parameters need to identify of the data source or one of the peers that are just connected to the overlay network.

The management protocol used by peers in order to organize themself tries to optimize the tree overlay network accordingly to the following rule: every peer gains a position into the structure that is a function of the peer available bandwidth but also a function of the below IP network characteristics in term of delay between peers.

Laboratories emulations show that the structure introduced delay are of the same magnitude of the optimal delay, hence no tedious effects are introduced by the software in the multi-hop overlay path. Moreover adopted metrics lead to the individuation and exploitation of faster overlay path.

P3P exhibits great resilient properties. Robustness is reached by introducing a Multiple Description Coding techniques. If each description could be delivered by an independent distribution tree the effect of a peer failure is only a temporally *quality of service* reduction that could be recovered in a few seconds.

Reconnection time is modeled and experimentally estimated accordingly to the theoretical values. It is obviously related to the experimented RTT and experimental measures show that its variable component does not affect the magnitude of the time period.

3. SENSOR NETWORKS CONTENT DELIVERY STRATEGIES

A wireless sensor network (WSN) is a particular wireless ad hoc network consisting of autonomous devices distributed in an area of interest, that cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations. The collected information has to be routed to one or more sinks, special access point that allows the final user to gather the collected data.

Sensor nodes can be deployed randomly in the physical area under exam or installed at deliberately chosen points. Nodes may change their initial position, either due to incidental effects or to move to interesting locations. Often a WSN is an heterogeneous network, where nodes may differ in the type of sensors, in power computation, or in the presence or absence of special hardware such as GPS. Network topology could be a simple single-hop network, where each node is able to directly communicate with the sink, or a multi-hop network; topology affects many network characteristics, such as latency, robustness and capacity. Topology is mainly dictated by the transmission range of each node, that defines its coverage area. Coverage area also influences the information processing algorithm chosen in the network and offers some kind of power saving by allowing the adoption of sleep modes for redundant nodes.

WSN could be used in many application area, including environment and habitat monitoring, health care applications, home automation, and traffic control. To reliable develop these many types of services a WSN has to face many challenges, such as being able to operate in harsh environmental conditions, managing nodes mobility, dynamic network topologies and possible nodes failure, working with heterogeneity of nodes.

Each node of a sensor network is typically equipped with one or more sensors, a radio transceiver or other wireless communications device, a small microcontroller, and an energy source, usually a battery. Nodes can be of variable size but typically they are very small, even like a coin. The cost of a WSN is similarly variable depending on the size of the sensor network and the complexity required of individual sensor nodes. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and bandwidth. Among these the most important one is the energy. Typically replacing batteries to the multitude of nodes present in the network is not practical and feasible; on the other hand many applications require the presence of long lifetime networks in order to be reliable as much as possible. All the layers of a classic protocol stack have to be redesigned with the purpose of energy saving, both by adding new paradigms and functionality to the existent protocols, by proposing new energy saving protocols and even by thinking to inter-layer solutions and adaptations.

3.1 Multisink routing

The primary function of a sensor network is the collection of sensor data and their delivery to the sinks, gateway nodes with more processing power that provide collected information to the final users. Telecommunication world is evolving towards the fourth generation systems, characterized by heterogeneous scenarios where different radio technologies and networks are expected to seamlessly interact together. By following this trend, the work presented in this Section considers a multi provider wireless sensor network, with more interconnection points (sinks) distributed in the whole sensing area. Another typical aspect in sensor networks concerns the need to add or upgrade the software running on sensor nodes without having to physically reach each individual device, or the need to give some commands to a particular group of nodes acting also as actuators interacting directly with the environment (e.g., temperature conditioning). Both the two situations require the need of a communication from a source to multiple destinations. Many existing works about wireless sensor networks deal with the problem of delivering information between sensors and a single sink; this section is related on a scenario where collected data have to be delivered to a group of sinks distributed in the whole area and interested to a particular collected data type (temperature, lightening...). The same approach could be followed to perform the complementary task, i.e., the communication between a sink and more sensor nodes.

A common approach to disseminate data in wireless sensor networks is based on geographic routing solutions, i.e., routing protocols that thanks to nodes location information route packets geographically towards the destination by forwarding to an awake neighbor node that is located furthest towards the destination, in order to achieve energy efficiency as much as possible. Many variants of geographic routing protocols have been proposed [15] [16] [17] [18] [19]: are based on the knowledge of the nodes locations and on the definition

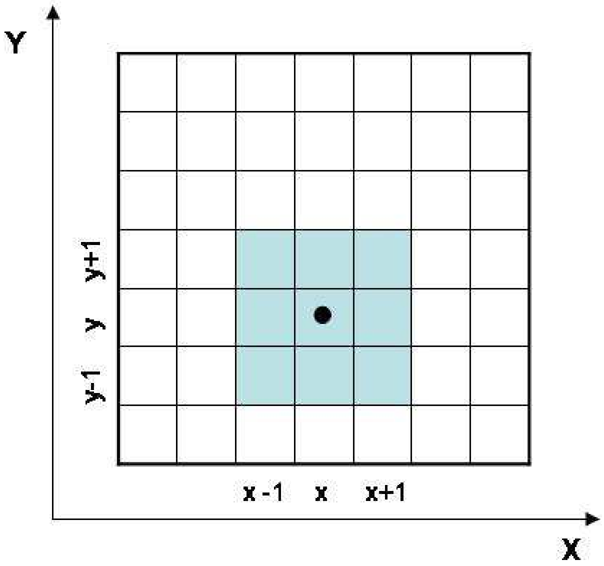


Fig. 3.1: Grid scenario.

of the distances between nodes in order to select the next forwarder.

This section proposes a data dissemination protocol for multisink wireless sensor networks, based on a *colored area* concept related to the nodes coverage area and on geographic multiple sinks locations. This allows to investigate how the inter-nodes distance definition could affect the protocol performance; in particular two distance metrics (the classic Euclidean and a metric that takes into account the nodes radio signal coverage) have been introduced; these can foresee different number of transmissions required to perform a task, giving different routing path selections and different energy consumption expectations.

3.1.1 Dissemination protocol

The considered sensor network is deployed in a square area of side L. As in [17] the whole area is divided in L^2 square regions, called grids. Grid can be treated as virtual macro-nodes: all nodes in the same grid can be interchangeably used for routing purposes. In order to ensure connectivity, the hypothesis to always have at least one awake node in each grid has been taken; the awake node among all the nodes in the same grid changes at different instants, since typically in sensor networks nodes switch between sleep and awake state. This condition does not need a global knowledge of the state of the network but only a knowledge limited to a local area of a single grid. As shown in Figure 3.1, the transmission range of a sensor node is equal to nine grids, i.e., a packet broadcasted by a node inside the grid (x, y) is received by all awake nodes inside grids $(x+i, y+j)$, with $i, j \in \{-1, 0, 1\}$; two nodes in neighboring grids are in transmission range of each other. A group of S sinks placed in non overlapped grids positions $pos_i = (xs_i, ys_i)$, with $i = 1 \dots S$ are interested in recovering data from the region (x_0, y_0); therefore a sensor source placed in the grid (x_0, y_0) that has to deliver data to all the S sinks. Suppose that all the nodes in the network are aware of their position and of the positions of all the sinks; a lot of solutions exist to ensure this assumption [20] [21]. The term *colored area* means the total amount of grids reached by data in a certain moment; for example, in Figure 3.1 when node inside grid (x, y), represented by the black point, broadcasts its data, the colored area is equal to the nine colored squares $(x+i, y+j)$ with $i, j \in \{-1, 0, 1\}$. The data dissemination protocol taken into consideration works as follows: starting from the source and until all the S sinks have been reached, data are sent to the nearest among all the not already reached sinks; this follows the principles of the nearest neighbor algorithm solving the traveling salesman problem [22]. Data delivery goes on in a multi hop way, since as mentioned in the previous hypothesis nodes coverage area is equal only to the node grid and the eight grids around it. At every hop, the next forwarder can be chosen among all the possible nodes inside the already *colored area*, that are grids that already own data. For example, suppose that three sinks are present, S_1, S_2 and S_3. The data dissemination process begins from the source, that broadcasts data into its coverage range, so a first coloration of nine grids takes place. The first destination sink is chosen by calculating the couple at minimal relative distance between all the possible couples (S_i, grid in the *colored area*), with $i = 1 \dots S$. Once a sink has been selected (e.g., S_1), data are transmitted to it by electing at each hop as next forwarder the node inside the present *colored area* that is nearest to the selected sink S_1, with respect to the selected metric. The chosen forwarder broadcasts the received data in its coverage area, so the *colored area* progressively expands; the process continues in the same way until data reaches S_1. Then another sink among the not already reached ones is chosen in the same way and the process continues until all the sinks have been reached. This procedure has been chosen since it is simple enough to be implemented in wireless sensor devices, usually equipped with limited processing and power resources.

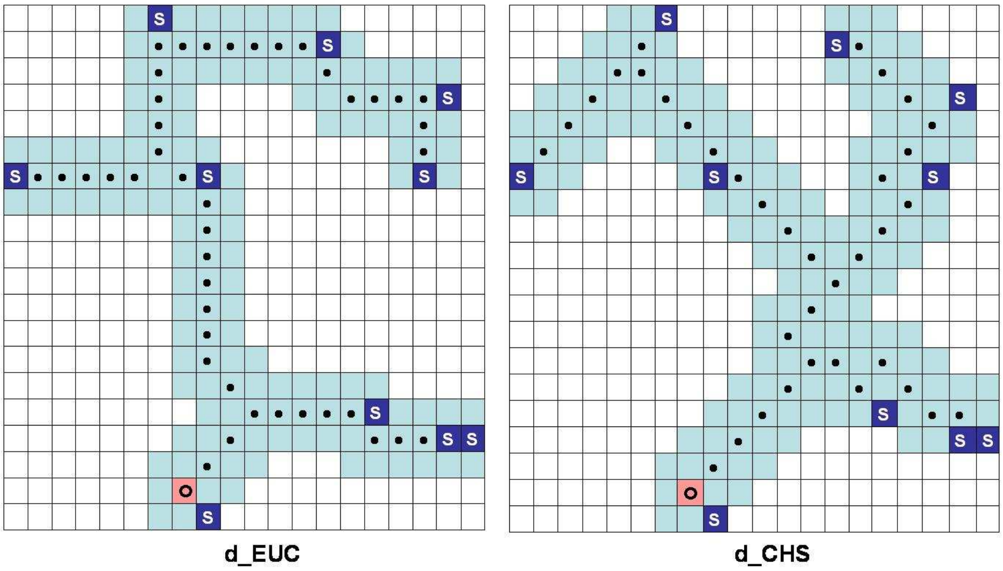


Fig. 3.2: Data dissemination protocol. The area side is $L = 20$. The source is in grid identified by the O, the $S = 10$ are in the darkest grids with capital S. *Colored area* is identified by the colored grids and the chosen forwarder nodes are in grids with black points. The Figure in the left refers to the protocol with d_{EUC} metric while the Figure in the right refers to the d_{CHS} metric.

3.1.2 Distance metrics definition

A central point is how to calculate the nearest sink and chose the next node forwarder. This measurements is based on the distance between two points in the area. The presence of the grid could make several distance metrics possible which give different answers for the distance between the same pairs of points. Consider the following two definitions:

Euclidean distance: this is the classic straight line distance between two points; if the two points have coordinates (x_1, y_1) and (x_2, y_2) than the Euclidean distance is given by:

$$d_{EUC} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Chessboard distance: this metric assumes that you can make moves on the grid as if you were a king making moves in chess, i.e., a diagonal move counts the same as a horizontal move [23]; this metric takes into account the electromagnetic behavior of the radio signal and the coverage area of a sensor node; its physical meaning derives from the consideration that a packet broadcasted by a sensor in grid (x, y) is received in all the grids

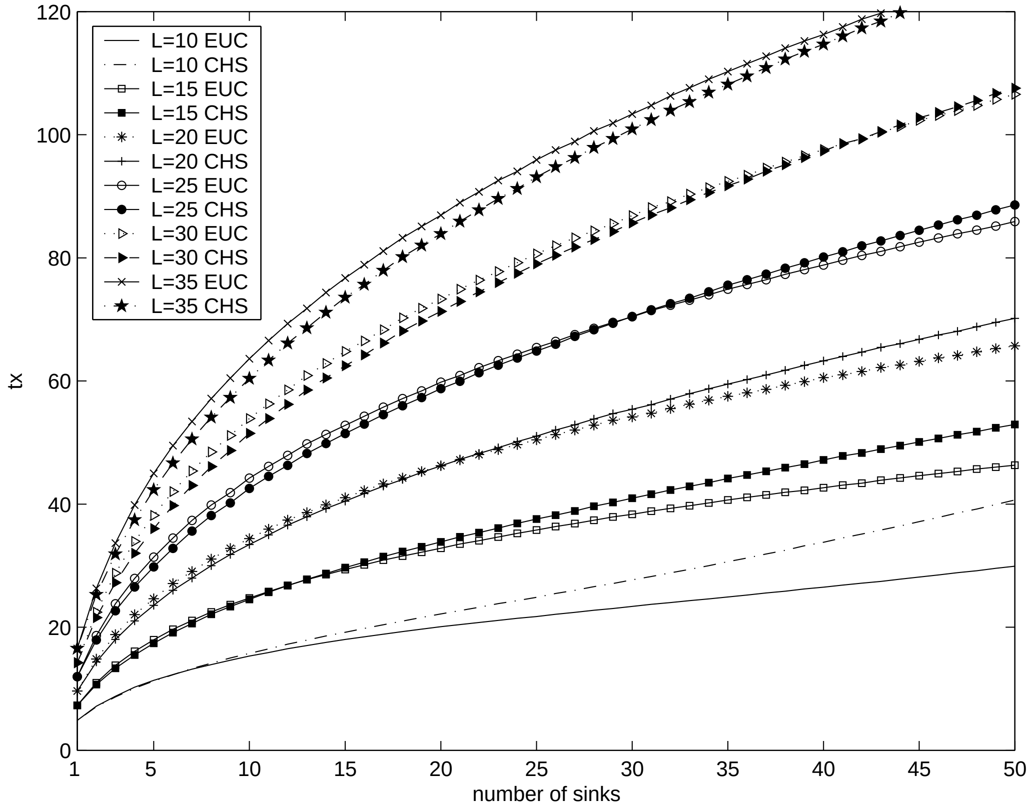


Fig. 3.3: Number of transmissions needed to reach all the S sinks.

$(x \pm 1, y \pm 1)$, so in one single hop the packet is received both from one hop neighboring horizontal grids $\{(x, y \pm 1),(x \pm 1, y)\}$ and from one hop diagonal grids $(x \pm 1, y \pm 1)$; this means that the distance is given by:

$$d_{CHS} = \max(|x_2 - x_1|, |y_2 - y_1|)$$

Figure 3.2 shows the dissemination protocol and the progressive *colored area* expansion, in case of the two defined distance metrics.

3.1.3 Comparisons and results

Forwarder nodes selection and routing energy consumption are strictly related to the number of packet transmissions. Different definitions of the inter-nodes distance could affect the data dissemination protocols in terms of number of transmissions required to perform a task. The task considered in this work is the distribution of the data from a source to a subset S of sinks deployed in the whole area. This aspect has been evaluated through

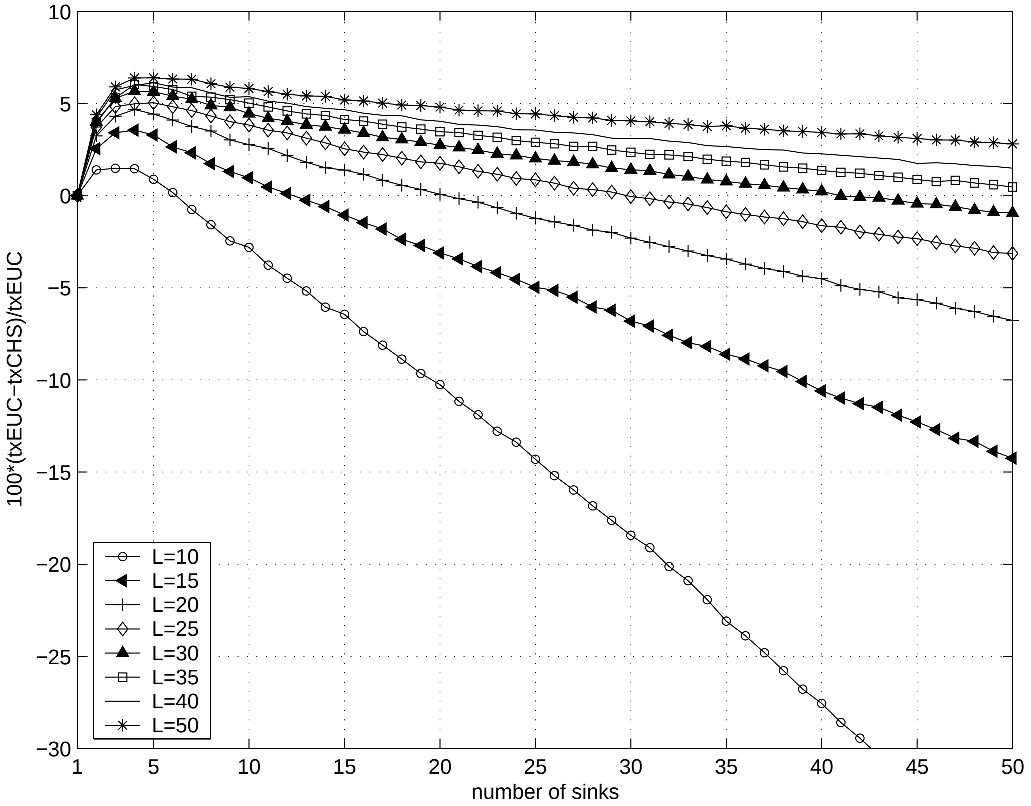


Fig. 3.4: Percentage difference of number of transmissions of the two metrics with respect to the classic *Euclidean distance*.

a specific simulator written in C: it allows to set the environment (L, S), it randomly deploys the S sinks and the source and it calculates the average number of transmissions needed to deliver the data to all the S sinks with both the two distance metrics. In Figure 3.3, the average number of transmissions is plotted as a function of the number of sinks S, by varying S from 1 to 50; also very big values of S have been included with the aim to investigate also very stressed scenarios, even if the most practical ones are characterized in general with values of S less than 20; different area size ranging from $L = 10$ to $L = 35$ are plotted together in the same Figure in order to jointly investigate the impact of the distance metrics and of the sinks density. Since with d_{CHS} a diagonal one hop move counts the same as a horizontal one hop move, at the first sight this metric seems to be always the best, in terms of minimal number of transmissions needed to perform the task. This is not always true. In fact, by focusing the attention on a specific scenario with a fix L, the curves related to the two different distance metrics show an intersection: more specifically, for smaller values of S the number of transmissions with the *Euclidean distance* are greater with respect to the ones with the *Chessboard distance* while for higher values of S the trend reverses. How evidenced in Figure 3.3, the crossing point between the curves related to the two mentioned distance metrics clearly depends on the (L, S) scenario; in particular, as the whole area side L increases the crossing point occurs with higher values of S. The ratio between the number of sinks S and the whole area dimension L^2 in correspondence of the interconnection points can be easily calculated: as the area side increases the crossing point occurs at lower sinks density values, ranging between 0.06 when $L = 10$ and 0.045 when $L = 30$. The reason why the crossing points occurs at these particular (L, S) scenario is due to the research algorithm applied: at every step the system searches for the couple of points at minimal relative distance and stops at the first one founded out, even if other couple with the same distance could be present; the choice of the couple between all the possible ones is invariant with respect to the selected sink under exam but is not invariant with respect to all the other not already reached sinks: in fact the choice of a particular forwarder affects the shape of the *colored area.* At higher sinks density the area of the *colored area* increases and this results in a greater number of possible couple with the same distance, so more possible different path selections. Nevertheless the mentioned protocol has been chosen since it is less complicated than an exhaustive research of all the optimal forwarders among all the potential couples.

It is interesting to calculate the percentage difference between the number of transmissions indicated by the two metrics with respect to the number of transmissions required by the classic distance definition d_{EUC}:

$$diff_{\%} = 100\frac{tx_{EUC} - tx_{CHS}}{tx_{EUC}}$$

Since the number of transmissions is directly linked to the sensors energy consumption this percentage difference can be seen a measure of the energy gain that could be obtained by choosing the right distance metric. Figure 3.4 shows this percentage difference plotted as a function of the number of sinks S and for different values of the area side, ranging from $L = 10$ to $L = 50$. The area of major interest, where $S \leq 20$, is characterized by a number of transmissions with the *Euclidean distance* that exceeds the number of transmissions with the *Chessboard distance*: the percentage difference in the smaller area with $L = 10$ is only about 2% but it increases with the area side and reaches values around the 7% when $L = 50$: so an energy gain around 7% is obtained thanks to the use of the *Chessboard distance* instead of the classic *Euclidean distance.*

3.2 M-GeRaf

Multisink GeRaf, [27] [28], is a declination of the GeRaf protocol ideas, [26], in multisink wireless sensor network environments.

It is based on geographic random forwarding routing and receivers competition realized by means dynamically computed back-off time.

In the following analysis sensor nodes may be stationary, densely deployed and randomly turn on and off, thereby providing a random topology. Each node has some knowledge of its own position and of the position of the sink nodes.

Likewise GeRaf once a node has to send a packet, it sends it specifying the list of the destinations. All the listening nodes in the coverage area receive the packet and schedule or not schedule the data forwarding after a back-off time.

Back-off time is calculated by evaluating the advancement toward destinations and the forward decision is taken evaluating the received destination set.

3.2.1 Design rationale

The generic situations, Figure 3.5, is the following: s is the actual source node, i.e., node that has to transmit data, and its position is $P_s \equiv (x_s, y_s)$. s data destination set is

$$D_s = \{P_{S_1}, \ldots, P_{S_{N_s}}\} \tag{3.1}$$

in which N_s is the current destination amount; s coverage area contains N awoken sensor that are able to forward the received packet and whose positions are $P_i \equiv (x_i, y_i)$, $i = 1 \ldots N$.

The subset of residual destinations of the node i can be defined by:

$$D_i = \{P_S \in D_s : d(P_S, P_i) < d(P_S, P_s)\} \tag{3.2}$$

If destination set is empty the packets will be discarded. Each node that receives a packet waits for a back-off time computed following equation (3.3).

$$T_b^{(i)} = k \cdot \min_{S \in D_i}\{\frac{1}{d(P_S, P_s) - d(P_S, P_i)} - \frac{1}{d(P_i, P_s)}\} \tag{3.3}$$

The triangle inequality asserts that awoken sensors in sink directions experiment no back-off time. Moreover, in order to avoid collisions, a little random

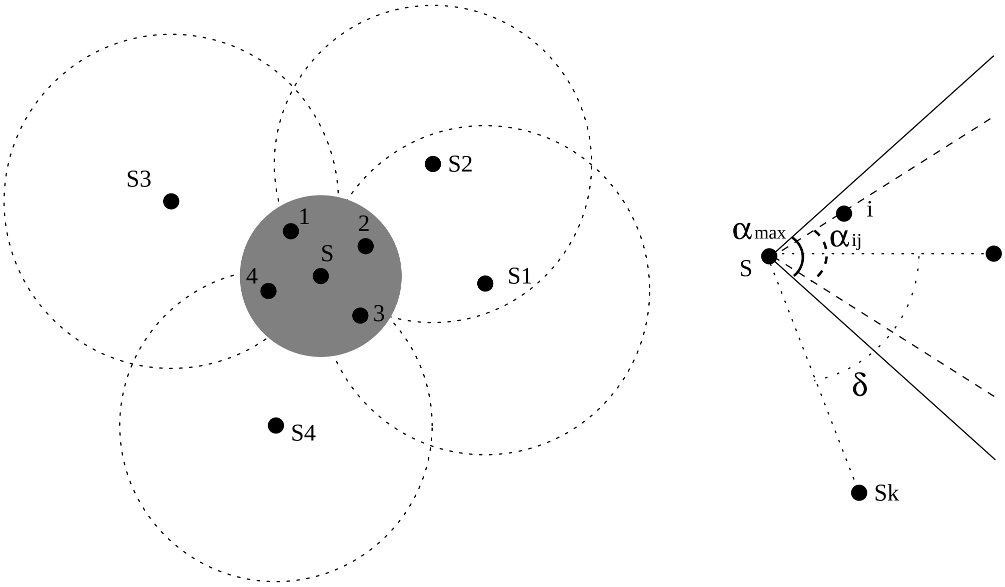


Fig. 3.5: Following eq.3.2, if $D_s = \{S_1, S_2, S_3\}$, $D_1 = \{S_2, S_3\}$, $D_2 = \{S_2, S_1\}$, $D_3 = \{S_1\}$, $D_4 = \{S_3\}$. On the right a graphical presentation of the maximum allowed apex angle, α_{max}, and of the the apex angle of the shrink-est cone in the S_j direction of that contains i, $\alpha_{i,j}$, are depicted

period is added to the calculated one:

$$T_{eff}^{(n_i)} = T_b^{(n_i)} \cdot (1 + \frac{rand()}{g}) \tag{3.4}$$

where $rand()$ is a random variable uniformly distributed in the interval $[0, 1]$ and $g \gg 1$.

Moreover, in order to avoid transmission and path duplications, awoken nodes in the first forwarder node coverage area remove the destination set elements of the forwarder node that have just transmitted the same data from their own destination set.

Unfortunately, not all the nodes that forward the received data to the same destination are in the same coverage area.
Nevertheless, by simple geometric considerations, it is possible to heavily reduce the amount of transmission of the approach indicated in the Equation 3.2 by shrinking α_{max}, the apex angle of the largest conical area (whose apex is P_s and whose axis reaches P_{S_j}) in which the receivers i accept destination S_j, see figure 3.5.

If $d(P_{S_j}, P_s) \gg d(P_i, P_s)$ hence

$$d(P_{S_j}, P_s) - d(P_{S_j}, P_i) \approx d(P_i, P_s) \cdot cos(\frac{\alpha_{i,j}}{2}) \tag{3.5}$$

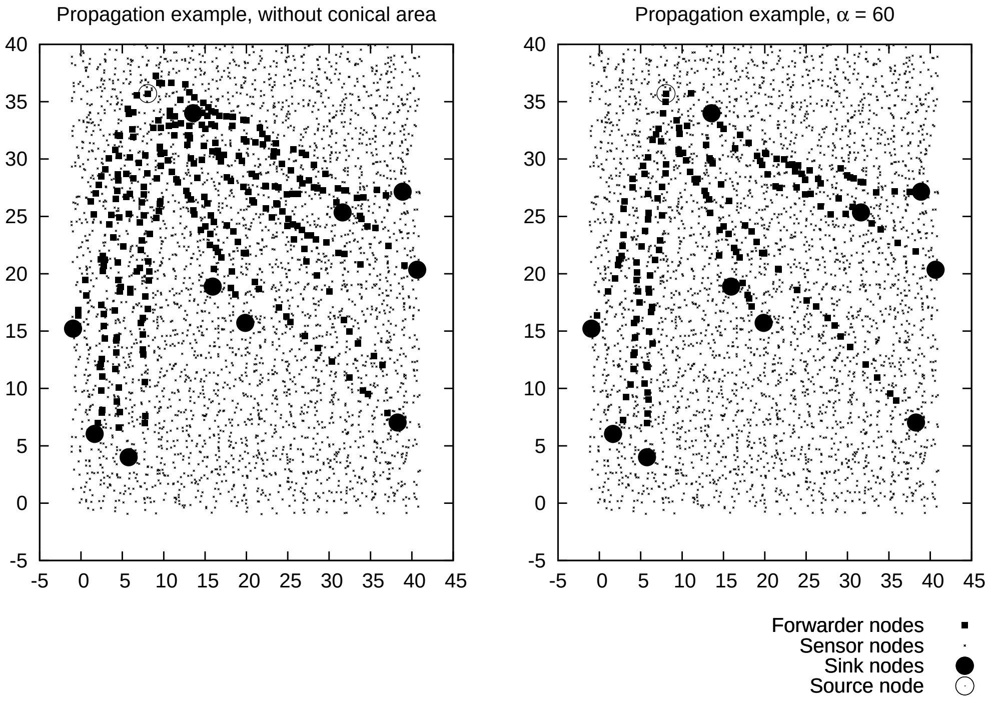


Fig. 3.6: In the right picture the performance improvement that can be obtained by setting $\alpha_{max} = 60^\circ$ in Equation 3.6, i.e., reducing conical decision area and the effect of the hidden terminal problem is depicted.

where $\alpha_{i,j}$ is the apex angle of the shrink-est cone in the P_{S_j} direction that contains P_i.

Therefore, in order to reduce the hidden terminal problem, each node that receives the packet calculates the its own destination set following

$$\begin{aligned} D_i = \{ & S \in D_s : \\ & d(P_S, P_s) - d(P_S, P_i) > d(P_i, P_s) \cdot cos(\tfrac{\alpha_{max}}{2})\} \end{aligned} \tag{3.6}$$

where α_{max} is the maximum allowed cone apex angle.

Decreasing α_{max} allow the designer to reduce the effect of the hidden terminal problem and, therefore, to reduce data path duplication and energy consumption.

In figure 3.6 is depicted how the hidden terminal problem could be overcome by reducing the apex angle of the conical acceptance zone, i.e., the hidden terminal problem triggers path duplication and hence increases energy consumption.

M-GeRaf does not require message loss recovery mechanism; if a packet will not be forwarded within a period, i.e., each node listens until someone which is nearer to the sink than itself forwards the packet, the transmission will be repeated.

3.2.2 Evaluation of the path bifurcation probability

In order to theoretically investigate the relation between path bifurcation probability and the maximum cone apex angle α_{max} the following hypothesis are assumed:

$$\forall S_j \; \forall i \;\; d(P_{S_j}, P_s) \gg d(P_i, P_s)$$

and that all the next hop nodes are at the same distance from the source s.

$$d(P_i, P_s) = d(P_j, P_s) = K \;\; \forall \; (i, j)$$

Therefore all the decisions will be based only on the $\alpha_{i,j}$ magnitude:

1. every node back-off time is related to the magnitude of

$$d(P_i, P_s) \cdot cos(\alpha_{i,j}/2) = K \cdot cos(\alpha_{i,j}/2)$$

2. and then only to $\alpha_{i,j}$:

$$\frac{\partial cos(\alpha_{i,j}/2)}{\partial \alpha_{i,j}} > 0 \;\; if \;\; |\alpha_{i,j}| < \pi$$

Moreover, in order to simplify the analysis, K is supposed to be the node coverage area radius.

In order to obtain path bifurcation probability values the following tools are introduced

- the matrix $\mathbf{A_{N \times N_S}}$, whose generic element $a_{i,j}$ is equal to $|\alpha_{i,j}|$
- the adjacency matrix $\mathbf{C_{N \times N}}$ whose generic element $c_{i,k}$ value is 1 and not 0 if and only if sensor k will be into the i node coverage area.

Therefore, given the sink and the neighbor awoken node positions, the conditional probability of n path bifurcations event during the next hop can be calculated.

This probability is the normalized frequency of the of the amount of next hop nodes minus one that will forward the received data; it is calculated as the amount of iteration of the following recursive deterministic model, in which $\mathbf{C_{*,i}}$ will be the $\mathbf{C}$ i-th column.

```
$k = 0$
$while(\min \mathbf{A}_{(k)} < \frac{\alpha_{max}}{2})$
  $m = rowindex(\min \mathbf{A})$
  $\mathbf{A}_{(k+1)} = \mathbf{A}_{(k)} + \mathbf{C}_{*,\mathbf{m}} \cdot \mathbf{D}^{(\mathbf{m})} \cdot \frac{\alpha_{max}}{2}$
 $k++;$
```

In detail

1. At first the algorithm checks if the minimum element of $\mathbf{A}$ is smaller than $\frac{\alpha_{max}}{2}$, i.e., it checks the existence of at least a forwarder node.
2. the algorithm defines that m is the row index of the minimum element of $\mathbf{A}$, i.e., m is the identifier of the first forwarder node.
3. the algorithm takes into account the target destination of node m transmission by introducing the row vector $\mathbf{D}^{(\mathbf{m})}_{\mathbf{1 \times N_S}}$ whose elements d_j are set to 1 (otherwise to 0) if and only if S_j is an m packet destination;
4. the algorithm removes node m packet destinations from the m adjacent nodes destination set by adding a large fixed amount to elements of the $\mathbf{A}$ rows that describe m adjacent nodes.

and so all the previous steps are repeated until point one fails. The amount of iterations, k, and the amount of the nodes that forward the received data are the same.

δ is defined as the angle between adjacent sink nodes conical area axes, see figure 3.5. In the following the behavior of the probability that a single packet will be forwarded n times is by varying the maximum allowed cone apex angle, α_{max}, and the angle between adjacent sink nodes, δ, is addressed.

We take into account node density introducing Δ, the average angle between next hop adjacent nodes $(i, i+1)$. If next forwarder nodes are close to the coverage area border, Δ will be intimately related to node density.

On the assumption that exist a references system of the sink node set and a references system of the next hop nodes set in which all element of the same system have got fixed position, but these two systems only rotate one respect to the other.

Furthermore on the assumption that the relative position of these two system is uniformly distributed over 2π radians angle, the required probabilities can be calculated normalizing the obtained frequency values.

Some quite obvious system behavior:

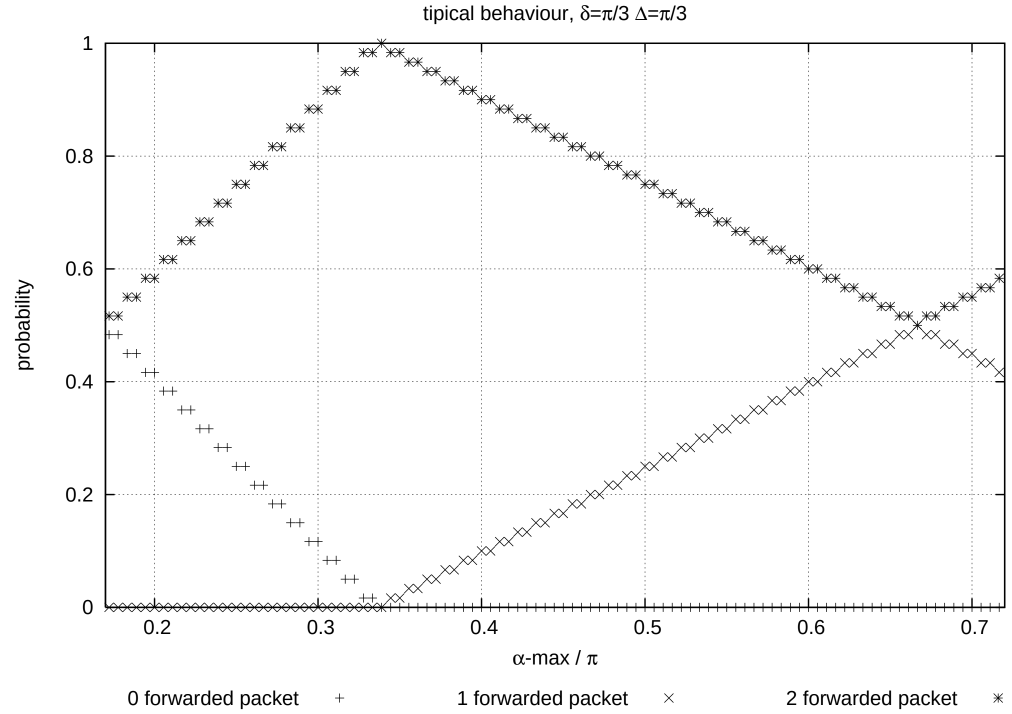


Fig. 3.7: In this theoretically based evaluation δ, the largest angle between a couple of sink nodes is $\pi/3$ radians; next hop nodes are $\pi/3$ radians spaced one from another too, $\Delta = \pi/3$. We point out that there is no need to consider more than 2 forwarded packet because, if $\max \delta \leq \Delta$ no more than two next hope node are involved in the packet forwarding issues.

- if $max(\delta) < \Delta$ over a subset of destination sink there is no need, in the evaluation, to consider other sinks than the farthest couple one.
- if $\delta > \alpha_{max}$ the greater-than-one outgoing packet rate can not be zero because data packets have to be forwarded by different nodes to different destinations due to the system path bifurcation rules.
- if $\delta > \pi/3$, due to the simple coverage area model adopted, i.e., each node coverage area is a circular area of the same measure of the other ones[1], the probability of at least a path bifurcation is one.

In Figure 3.7 the amount of nodes in the s coverage area, N, is 6 and the forwarder nodes are dislocated likewise hexagon apex, i.e., $\Delta = \pi/3$; moreover the two farthest sink are $\pi/3$ radians one from another, $\delta = \pi/3$. We can appreciate some typical system behavior:

- if $\alpha_{max} < \Delta$, i.e., each couple of adjacent nodes are $\pi/3$ radians separated, there is a non-zero probability of do not forward any packet towards a selected sink direction.
- if $\alpha_{max} < \delta$, an excessive duplication rate due to a small α_{max}, i.e., too tight allowed destination cone areas, can be experimented.
- if $\alpha_{max} > 2\pi/3$ the bifurcation rate increases. Due to the simple adopted geometry only the nodes located in the adjacent hexagon apexes are in the same coverage area. Hence, if the allowed conical sector include nodes that are not in the same coverage area, a non-zero packet duplication rate can be exhibited.

In Figure 3.8 the influence of an increasing sensor density is showed. An higher density leads to, fixed δ and α_{max}, an higher probability of multiple transmission event to different destinations and, often, to the same destinations.

This last behavior, in which more than one node, one not in the coverage area of the others but both in the source node coverage area, forwards the received packet to the same destinations have to be avoided in order to reduce the average energy consumption.

Previous considerations lead to heuristically calculate an optimum α_{max} as a function of the only node density and to the geometrical assumption of

[1] if a node is close to the end of the coverage area the angle subtended at the center of the circle (which is the location of the source node) of its coverage area is $2\pi/3$, hence a node can listen another one if and only if the angle at the center between them is smaller than $\pi/3$.

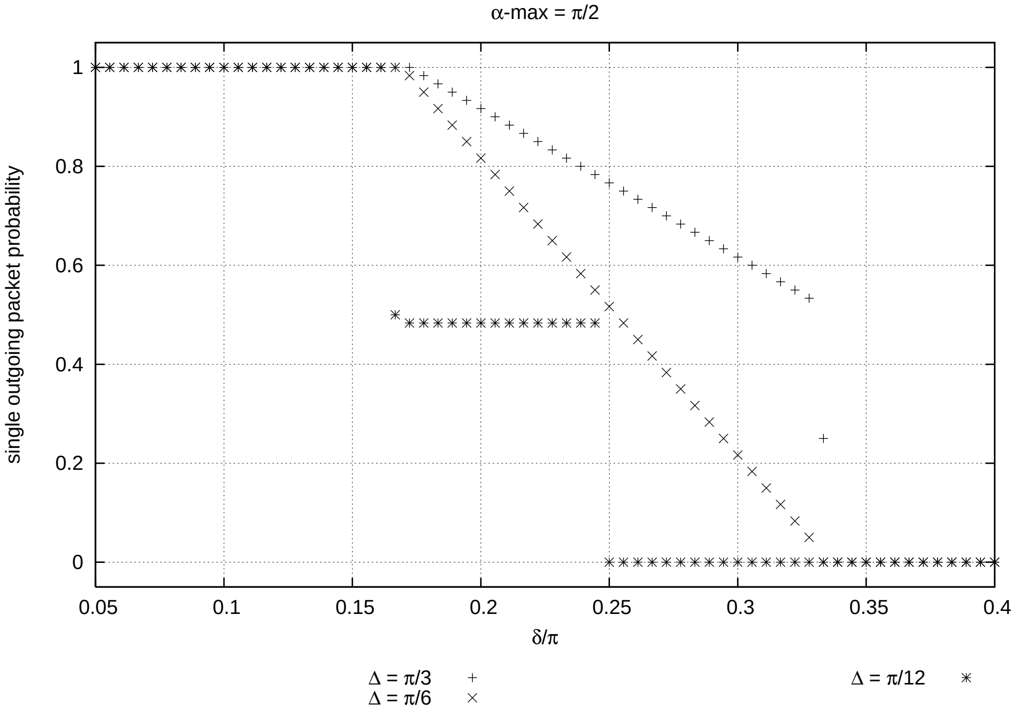


Fig. 3.8: In this theoretically based evaluation α_{max} is fixed to $\pi/2$, but the single forwarded packet probability are depicted using three different node density, $\Delta = \pi/3$, $\Delta = \pi/6$, $\Delta = \pi/12$

circular coverage area:

$$\alpha_{max}^{opt} = 60^\circ + \Delta \tag{3.7}$$

where Δ is a density approximation and could be approximated by the ratio between 360° and the amount of sensor in the coverage area.

Adopting the $\alpha_{max} = \alpha_{max}^{opt}$ rule all the node that can accept a particular destination are, on average, in the same coverage area, therefore, the probability of obtain multiple forwarded packets to the same destinations through different path is minimized.

Nevertheless α_{max}^{opt} is the largest angle that minimizes the bifurcation probability therefore the one that minimizes the probability of reach an empty conical zone, i.e., no forwarded data to a particular destination.

The existence of the optimum of α_{max} related to the node density is supported by simulation: as depicted in picture 3.9, α_{max} values smaller than the optimal one does not lead to system performance improvement. Moreover the transmission amount exhibits a limit value that is very close to the one obtained adopting the analytically optimal values of α_{max}. Therefore the angle α_{max}^{opt}, theoretically calculated, could be considered a pivotal parameter that allows the system designers to minimize the amount of transmissions.

3.2.3 Data delivery improvement

Adopting a value of α_{max} significantly smaller than the *local* α_{max}^{opt}, i.e., the one obtained measuring the effective local node density and not the global one, leads to the possibility of not reach all the destination nodes.

The effects of a small α_{max}^{opt} over the system performances has to be investigated: in real environments, due to battery lifetime issues or to propagation issues, experimented local node density can change rapidly and this misconfiguration is not unusual.

In figure 3.10 is shown this misconfiguration effect on system behavior. After removing the grid assumption, i.e., at least one awoken node for each grid, and therefore specifying only the global area node density, the system performance fall down. Picture on the left compares P_{all}, the probability of reach all the sink node without retransmission, i.e., without waiting for a topology change. Picture on the right shows TA_{all}, the product between the effective average node transmission amount and P_{all}^{-1}. TA_{all} estimates the amount of transmission required to reach all the sink nodes even when some sinks could not be reached by the transmitted data. The amount of effective transmissions is quite the same but the P_{all} reduction produces an increment of TA_{all} values.

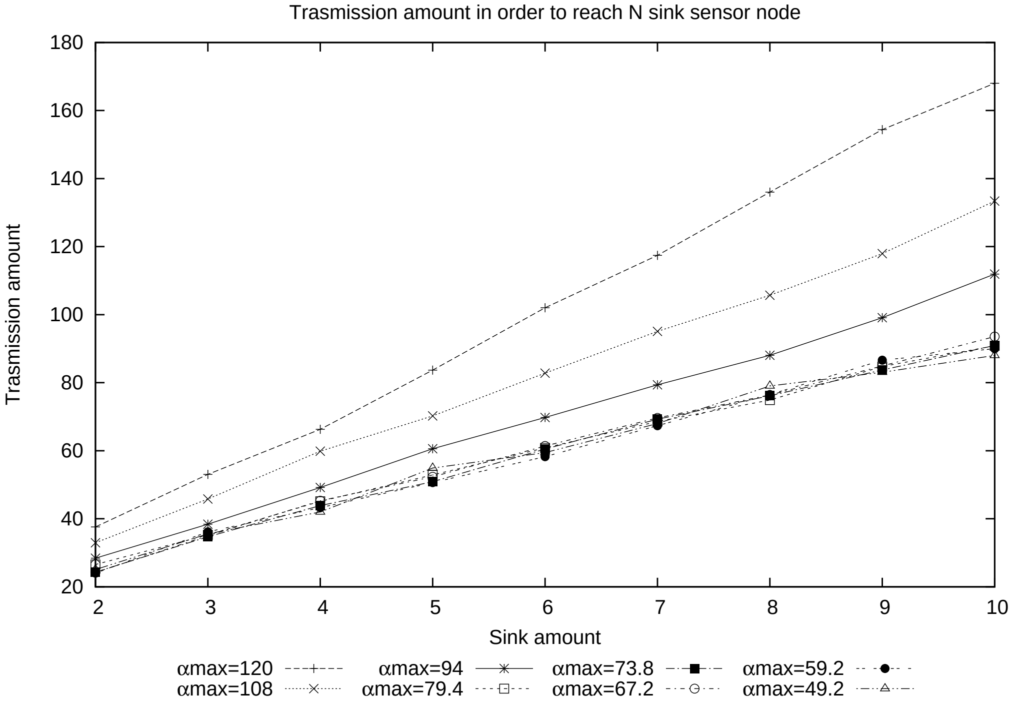


Fig. 3.9: Effect of different α_{max} on the amount of needed transmissions to reach n sink with a fixed sensor density. Each node coverage zone contains about 48 sensor nodes, therefore $\Delta \simeq 7.5°$. The transmission amount decreases until $\alpha_{max} > \alpha_{max}^{opt} = 67.5°$.

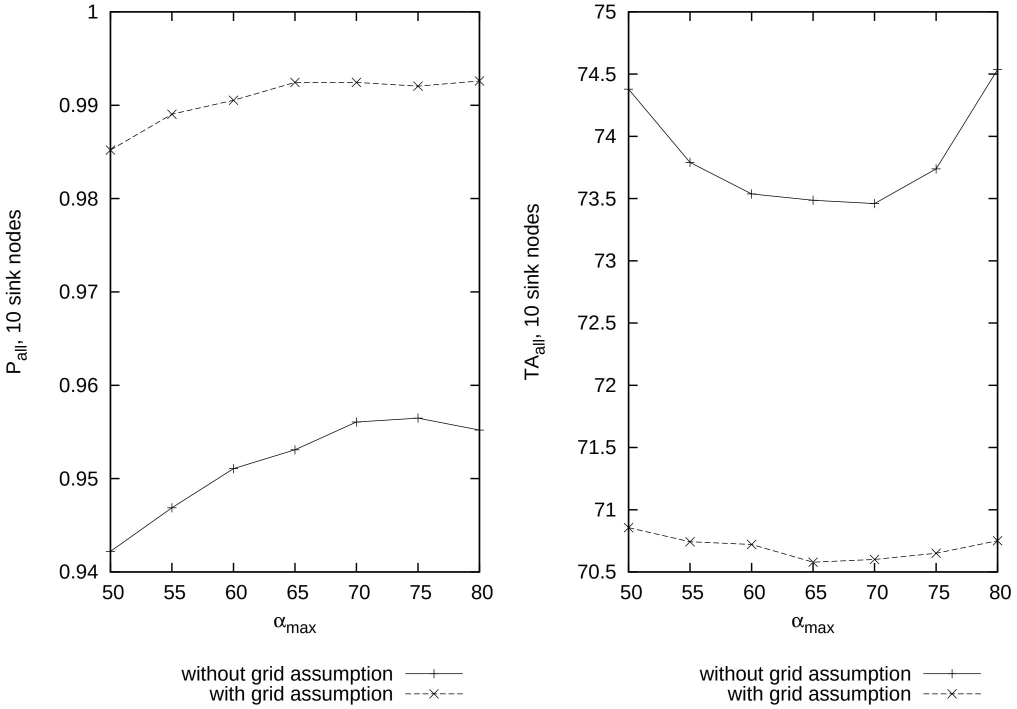


Fig. 3.10: Performances of the M-GeRaf system with and without grid assumption: probability of reach all the destination nodes exploiting only the awoken nodes and average transmission amount required to reach all sink nodes, $\Delta \simeq 15°$

In order to reduce the performance degradation shown in figure 3.10 a new retransmission algorithm that could increases P_{all} has been developed.
This approach introduces a retransmission schema that not entrust only in topology change but also in a conical area enlargement : nodes that receives multiple copies of the same data from the same source calculate a source acceptance area larger than the previous transmissions acceptance cone and, hence, they potentially could receive and forward the repeated data.

This algorithm coexists and not substitutes the original retransmission algorithm that schedule another transmission if, after a timeout expiration, any nodes has not repeated the data transmission yet.

Original algorithm supposes that some network topology variation happens between the first transmission and the following ones but this approach supposes also that some awoken nodes exist close but not into the conical acceptance area and that they can be used as forwarded node if no one forwards the data before.

Different algorithm that enlarges the α_{max} after the first data transmission if no nodes forward the data has been developed. The forward decision is taken on the receiver side, therefore each nodes in the current source coverage area have to recognizes that the received packet is a repeated transmission and will calculate if they are or not in the enlarged source acceptance conical area.

In the left picture of figure 3.11 is plotted P_{all}, the probability of reach all the sink node varying the amount of sinks: P_{all} does not change in accordance to the amount of destinations but is only a function of α_{max}. In the right picture of figure 3.11 P_{all} fall down if $\alpha_{max} < \alpha_{max}^{opt}$ but it is almost the same if $\alpha_{max} > \alpha_{max}^{opt}$. Adopting the new algorithm P_{all} is close to 1 until α_{max} is significantly smaller than its optimal value.

Therefore this behavior assures an improved resilience against local node density variation.

This approach reduces system delay and allow node to avoid to waiting for variation in network topology in order to continue the data delivery; moreover it increases the probability of reach all the sink node without network topology change.

In figure 3.13 is shown the effect of the proposed strategy together with the one in section 3.2.4 into a more general environment in which any grid assumption can not be supposed.

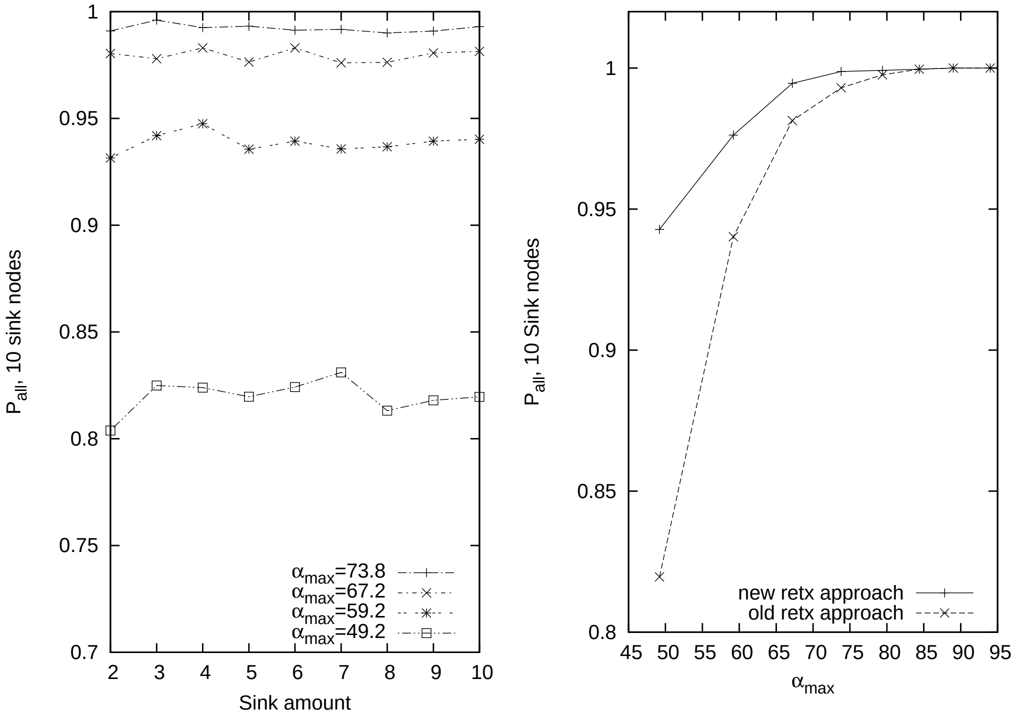


Fig. 3.11: Effect of new retransmission schema over transmission amount and probability of reach all the sink. Each node coverage zone contains about 24 sensor nodes, therefore $\Delta \simeq 15°$.

3.2.4 Metric improvement

Back-off time evaluation is pivotal in order to identify sensor whose position produces the larger advancement toward destinations and hence that has transmit the received packet before other nodes. Every sensor that has to forward the received data listens if another one transmits before its own back-off time expiration and, if so, it does not repeat the transmission, i.e., short back-off time should always mean good position advancement and also an overall transmission amount reduction.

The original evaluation, equation 3.3, focuses on the direction of sink nodes in order to increase or decrease the back-off time: by means triangular inequality each node evaluates a back-off time that decreases as soon as its own position is close to the lines identified by the actual source node point and each of the destination node points.

Adopting equation 3.3, nodes in the same direction but located at different distances from the actual source node calculate quite the same back-off time.

Moreover, when the distance from actual source node increases the back-off time evaluation function exhibits a floor that does not aid the best sensor position individuation.

In order to improve the system performances the magnitude of the back-off time gradient has to be significantly greater than 0 over all the conical acceptance area. Therefore the following back-off evaluations that will take into account a strong dependence on the magnitude of the advancement toward destination is introduced

$$T_b^{(i)} = k \cdot \min_{S \in D_i} \{ \frac{d(P_S, P_i)}{d(P_S, P_s)} \} \tag{3.8}$$

where $k = sup_{s,D_i}\{T_b^{(i)} \; \forall D_i, P_S, P_i, P_s\}$.

In equation 3.8 there is no dependences from advancement direction: direction decision will be completely delegated to the definition of the conical acceptance zone.

Both in equations 3.3 and 3.8 the back-off time is a function of only one destination, i.e., the one that minimizes back-off formulation.

This approach is typical of a greedy algorithm but does not consider other destinations. Our rationale guidelines is that nodes with larger destination set than the other ones, i.e., node in a strategical position, should experiment smaller back-off time than the other ones.

Therefore in order to take into account more than one destination a new

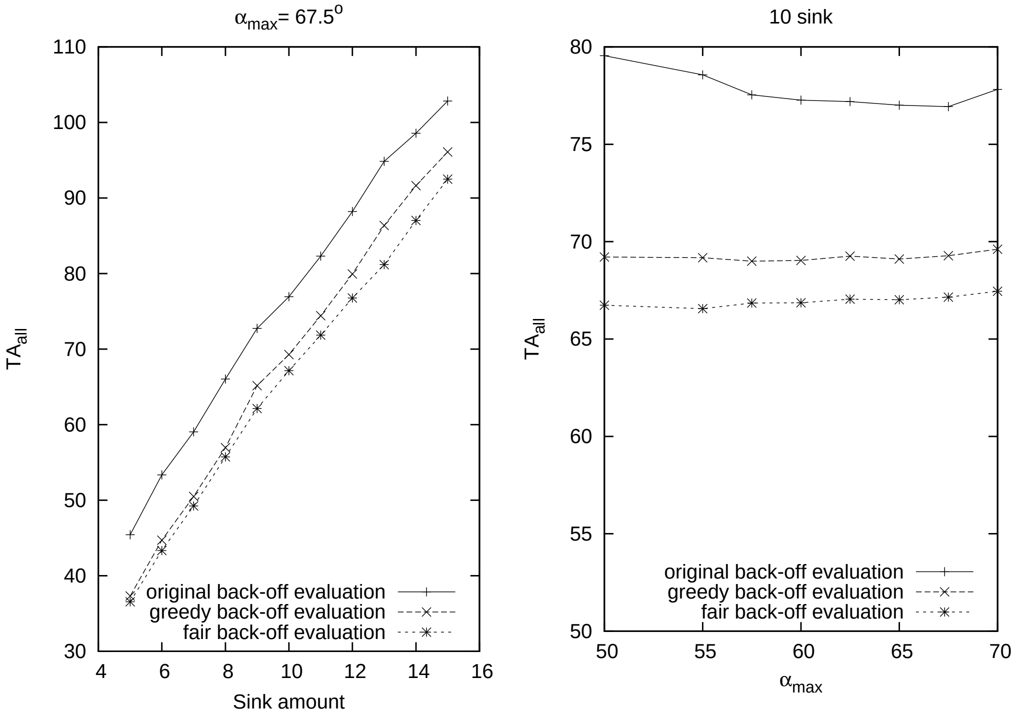


Fig. 3.12: Effect of equation 3.8 back-off evaluation over transmission amount and probability of reach all the sink. In this picture each node coverage zone contains about 48 sensor nodes, therefore $\Delta \simeq 7.5°$ and $\alpha^{opt}_{max} = 67.5°$

back-off time evaluation is introduced

$$T_b^{(i)} = k \cdot \frac{\min_{S \in D_i}\{\frac{d(P_S,P_i)}{d(P_S,P_s)}\}}{(\sharp D_i)^2} \tag{3.9}$$

Equation 3.9 adds to equation 3.8 the dependence from the destination set cardinality, $\sharp D_i$.

In picture 3.12 is depicted the effects of different back-off time evaluation:

- the original one,
- the *greedy* one (equation 3.8),
- the *fair* one (equation 3.9).

The absolute amount of transmissions is heavily reduced by both the new approach, left picture of figure 3.12, but the *fair* one performances are slightly better than the others. Variations in the amount of destination nodes does not affect significantly this improvement.

Moreover if the system behavior is not analyzed only for a range of α_{max} values in an around of α_{max}^{opt}, the new metrics will exhibits greater improvement for each α_{max} values.

Furthermore the new retransmission approach reduces the raise of TA_{all} when $\alpha_{max} < \alpha_{max}^{opt}$.

In figure 3.13 is depicted the performance of the original M-GeRaf and of the same system after the adjustments of section 3.2.4 and section 3.2.3. New retransmission approach and new back-off evaluation together produce a significantly improvement of both P_{all} and TA_{all} also in a more general environment in which the existence of at least one awoken node in the close grid (grid hypothesis) can not be supposed and only some knowledge of the overall node density is assumed.

3.2.5 Other back-off metric approach

In order to introduce some dependency on advancement direction in the back-off time evaluation some direction dependencies by means triangular inequality are proposed

$$T_b^{(i)} = k^{(1)} \cdot \min_{S \in D_i}\{\sqrt{T_b^{(i,S,1)} \cdot T_b^{(i,S,2)}}\} - k^{(2)} \tag{3.10}$$

$$T_b^{(i,S,1)} = \frac{1}{d(P_S,P_s) - d(P_S,P_i)} - \frac{1}{d(P_i,P_s)}$$

$$T_b^{(i,S,2)} = (1 + m\frac{d(P_S,P_i)}{d(P_S,P_s)})^2$$

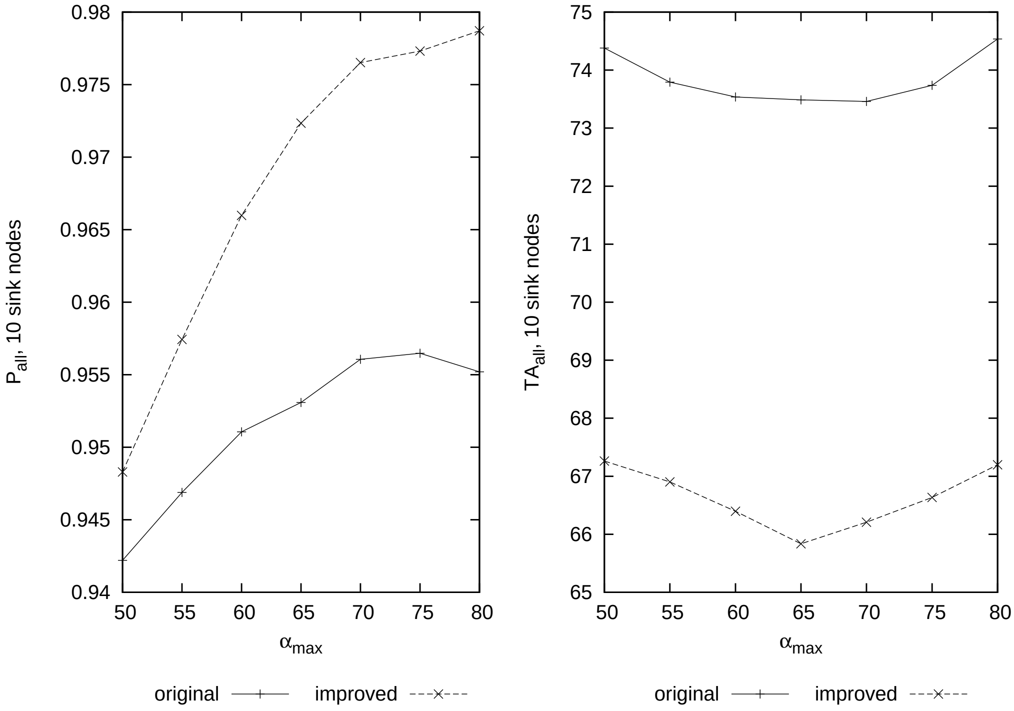


Fig. 3.13: Improvement of M-GeRaf performances without grid assumption

The value of $k^{(1)}$,$k^{(2)}$ are calculated in order to scale the calculated back-off time and in order to compare different evaluations with the same back-off time range values.

Nevertheless equation 3.8 outperforms the equation 3.10 results.

In order to introduce in the back-off time evaluation a stronger dependence from advancement toward each single node in current destinations set the back-off time could be defined as

$$T_b^{(i)} = k \cdot \frac{\sum_{S \in D_i} \left\{ \frac{d(P_S,P_i)}{d(P_S,P_s)} \right\}}{\sharp D_i} \tag{3.11}$$

where $\sharp D_i$ is the destination set cardinality. Unfortunately the amount of transmissions in order to reach all the nodes nearly double the equation 3.9 approach.

3.2.6 Position estimation error effect

A good position estimation is pivotal in M-GeRaf system: position information are used during the forward data decision process and during back-off calculation process.

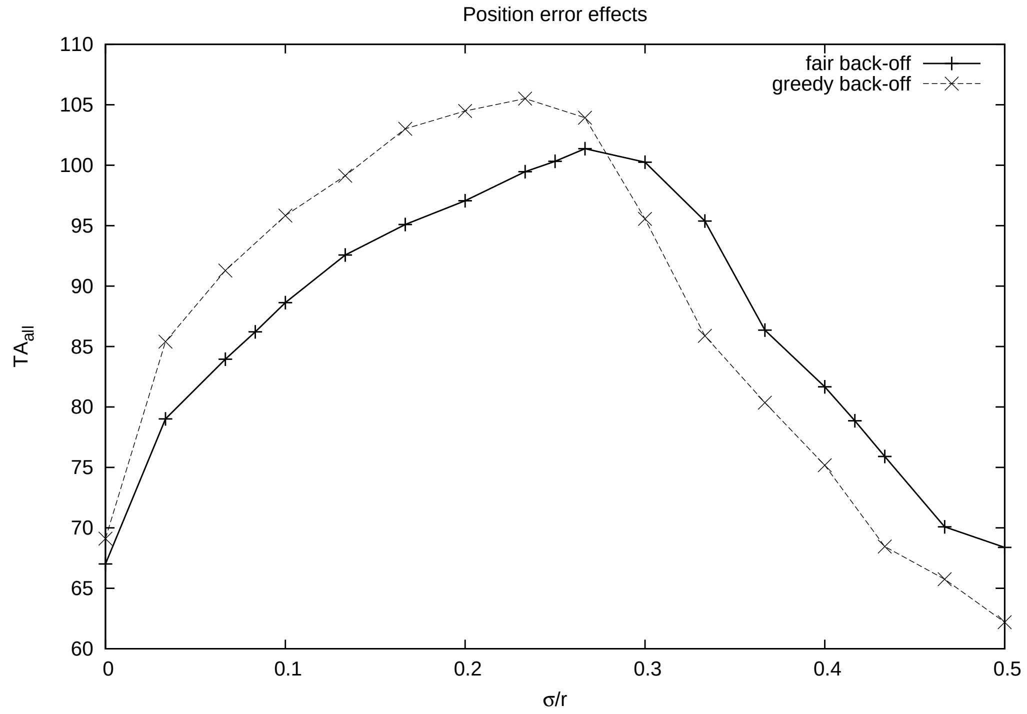


Fig. 3.14: Effect of increasing error standard deviation over transmission amount in order to reach 10 sensor sink nodes.

In order to take into account errors in the position estimation process a Gaussian additive error is added to the right position, therefore the system behavior is simulated varying the error probability standard deviation.

In picture 3.14 is depicted the amount of transmissions in order to reach 10 sink node varying the quotient between error standard deviation and coverage area average radius.

In Gaussian distribution about 95% of the values are within two standard deviations and about 99.7% lie within three standard deviations σ away from the mean μ.

Therefore the computed values close to $\frac{\sigma}{r} = 0.3$ are the ones in which nearly all the sensor nodes position estimation error magnitude is smaller than the coverage area; instead only the 95% of the sensor nodes position estimation error nearly the values computed close to $\frac{\sigma}{r} = 0.5$ are smaller than the average coverage area radius .

The system performance fall down if $\frac{\sigma}{r} > 0.3$: the probability of reach all the sensor nodes decreases rapidly and also, figure 3.14, the amount of transmissions in order to reach all the nodes, i.e., the forward decision process and the back-off time algorithm lose their own effectiveness.

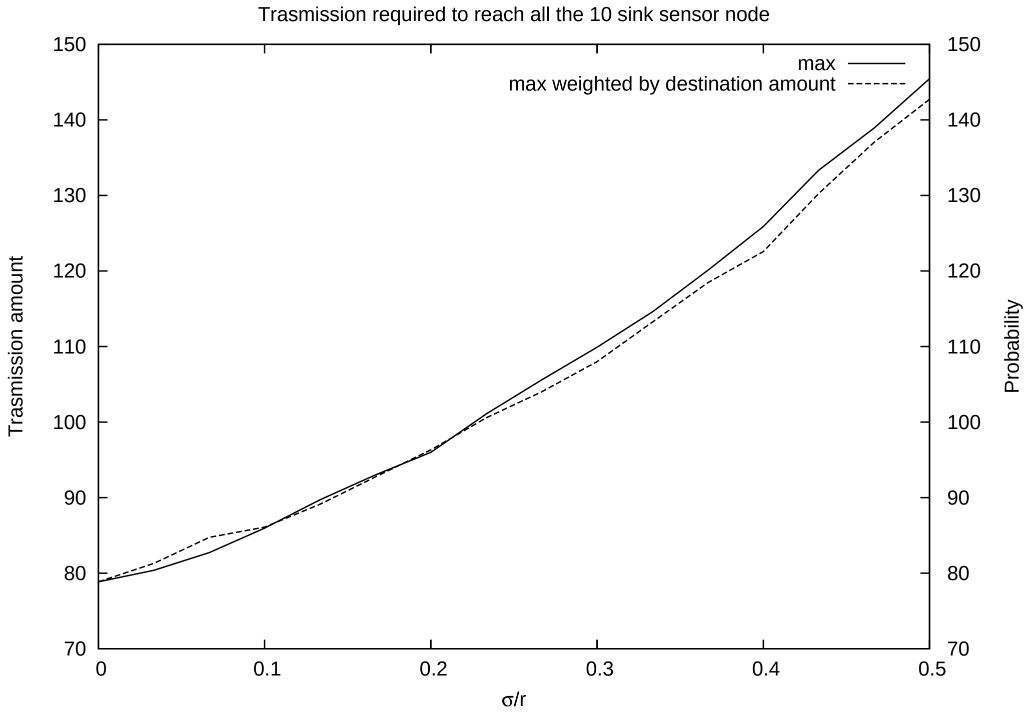


Fig. 3.15: Effect of increasing error standard deviation over amount of transmissions required to reach all the 10 sensor sink nodes

In figure 3.15 the average amount of transmissions required to reach all the sensor node increasing estimation error standard deviation are depicted.

The *fair* back-off time approach is more resilient to position error than the greedy approach: evaluating more than one destination helps the selection of node with lower position error.

3.2.7 Conclusion

M-Geraf, [27][28], is a data dissemination protocol for multisink wireless ad hoc sensor network that exploits random geographic routing approaches in order to achieve a reliable data delivery in networks with an aggressive power-off strategies. T

Reliability is achieved by exploiting the intrinsic broadcast characteristics of the wireless channel and the possibility of learning information from the neighbor transmissions.

The goal of M-GeRaf is to realize a new dissemination protocol that resolves efficiently the problem of reaching a set of destinations with only the knowledge of the own position and of the sink positions.

The analytical model results of bifurcation event probability developed in [27] are confirmed also in more general environment in which the grid assumption is removed, i.e., without next awoken node position assumptions.

Moreover the existence of an optimum apex angle of the conical decision area, α_{max}^{opt}, related to the greatest angle that minimizes the bifurcation event probability are confirmed not only by transmission amount optimization but also by the trend of the probability of reach all the sink sensor nodes.

M-GeRaf performances, in term of amount of transmissions in order to reach all the sink sensor nodes, TA_{all}, are heavily affected by the parameter α_{max} that have to be commensurate to the sensor node density in order to reduce the hidden terminal problem.

- If α_{max} is greater than the optimal one the amount of transmissions could grown and the global energy consumption too.
- If α_{max} is smaller than the optimal one the probability of reach all the sink nodes decreases, therefore, the transmission amount required to reach all the sink node, TA_{all} increases in despite of the effective amount of transmissions remains quite the same.

Local node density could vary significantly and hence global node density estimation could not be always the better choice. Therefore a new retransmission approach that possibly enlarges α_{max}, tries a retransmission and does not wait for network topology variation is introduced.

The effect of this algorithm is that, unlike the previous approach, the probability of reaching all the sensor node does not immediately fall down if $\alpha_{max} < \alpha_{max}^{opt}$. Nevertheless the new retransmission schema introduces a slight transmission amount increment due to the α_{max} sporadic enlargements.

This increment does not compromise system performance because the primary effect of this approach is the P_{all} improvement. Moreover P_{all} improvement leads to a more reactive data delivery, i.e., data that require a low transmission delay can be sent without waiting for network topology variation.

M-GeRaf introduces a contention scheme based on weighted back-off time, i.e., back-off time are calculated as a function of advancement toward a sink direction, that is designed in order to reduce the overall amount of transmissions.

The original back-off evaluation schema focuses on sink direction and does not take into account the magnitude of the advancement toward destinations. The second back-off evaluation, equation 3.9, overcomes this limitation and simulation tests show a great performance improvement in terms of transmission amount and, hence, energy consumption.

Moreover in the original adopted metric the first node that retransmits the data is the one with the greatest advancement toward direction of only one of the sink nodes, in this second metric the evaluation takes into account more than one destination and the cardinality of forwarder node destination set.

Finally the strength of M-GeRaf against position estimation errors is investigated. Position estimation is pivotal in order to take right packet forward decision and in order to calculate right back-off time.

An additive Gaussian position error is added and the error process power is increased. Simulation result demonstrates that the system performance fall down if the position estimation error could be greater than the average coverage area radius, i.e., nodes could lie over their own believed coverage area. This result stress the limit of the application of M-GeRaf protocol.

4. SECURITY IN CONTENT DELIVERY PARADIGM

A drawback of content delivery concerns their security. Crackers have found telecommunication networks and the Internet relatively easy to break into. Moreover the risks to users of wireless technology have increased exponentially as the service has become more popular. Free-space radio transmission in wireless networks makes eavesdropping easy and consequently, a security breach may result in unauthorized access, information theft, interference, jamming and service degradation. What makes it worse is that the sender and the intended receiver have little means of knowing whether the transmission has been intercepted or not, so the intrusion is virtually undetectable.

Cryptography can offer valid solutions to protect a network. Encryption can be used to ensure secrecy, by obscuring information to make it unreadable without special knowledge, but other techniques are still needed to make communications secure, particularly to verify the integrity and authenticity of a message; for example, a message authentication code (MAC) or digital signatures. Encryption or software code obfuscation is also used in software copy protection against reverse engineering, unauthorized application analysis, cracks and software piracy used in different encryption or obfuscating software. Encryption could therefore be an useful tool to protect communication in a content delivery network. This chapter discusses a Wide Trail Strategy based method able to enhance the security offered by an encryption scheme developed in the University of Ferrara and based on an extended Feistel structure.

4.1 Application of wide trail strategy to an extended Feistel cryptosystem

In [30], Filippini proposes a new scheme of iterated block cipher based on an extension of the Feistel structure: in this algorithm the round functions f operate on a more grained subblock and both the keys generation and the round functions are obtained from the quantized trajectory of a dynamical system with characteristics similar to chaotic systems; these cryptofunctions

f provide good diffusion and strength against differential attacks. Many studies related to the use of chaotic dynamic systems in cryptography were proposed in literature; chaotic systems have many interesting features (sensitivity to the initial condition, ergodicity and mixing properties) that may be linked to confusion and diffusion [31], the two basic concepts about cipher introduced by Shannon. In [32] authors show that several chaos based block ciphers do not behaves worse that the standards ones, therefore opening a novel chaos based approach to the design of block encryption schemes. In [30] a comparison with DES-like cryptosystems shows the effectiveness of Filippini's proposal in terms of minimum number of steps required to break the system. This chapter introduces a method based on the Wide Trail Strategy [33], that can greatly improve the robustness of the Filippini system against Differential and Linear attacks by increasing its diffusion characteristics and that outperforms the performances of the original Filippini's scheme.

4.1.1 Block Encryption algorithm

Let A and B denote finite alphabets of symbols. Let A^* denote the set of all strings that can be made with the elements of A (similarly for B^*). The set of all finite strings with length n over A is denoted by A_n.

Any one-to-one mapping

$$E : A^* \longrightarrow B^* \tag{4.1}$$

is called cryptographic transformation.

A string p in A^* that one wants to encrypt is called a *plaintext*. The result $c = E(p)$ will be called a *cyphertext*. Since E is a one-to-one mapping, its inverse must exist. It will be denoted by D and

$$D(E(p)) = p \; \forall p \tag{4.2}$$

A block-encryption crypto-system consists of a set of cryptographic transformations

$$E = \{E_k | k \in K^m\}\,,\; E_k : A^m \longrightarrow B^m \tag{4.3}$$

and corresponding set D. The index set K^m is called the key space, and an element in K^m is a *key*. m is the length of a data-block. Quite often $A = B = K = \{0, 1\}$ and $m = 64$ or $m = 128$.

A crypto-system can be viewed as a class of algorithms, indexed by a key.

Two general principles which guide the design of practical ciphers are *diffusion* and *confusion*.

- Diffusion means spreading out of the influence of a single plaintext or key digit over many ciphertext digits so as to hide the statistical structure of the plaintext.
- Confusion means use of transformations which complicate dependence of the statistics of ciphertext on the statistics of plaintext.

Most ciphers achieve the diffusion and the confusion by means of round repetition:

$$\begin{array}{lcl} x(0) & = & p \\ x(t) & = & E_k[x(t-1)]\ t = 1 \ldots r \\ c & = & x(r) \end{array} \tag{4.4}$$

4.1.2 Uniform and Feistel transformation

Block cipher transformation can be subdivided in

- Uniform networks (also called substitution-permutation networks): block ciphers in which every input bit is treated in a similar way. Some examples: SAFER, SHARK, Rijndael, and 3-WAY.

  An advantage of this approach is the inherent parallelism, while a disadvantage is that inverse algorithm (which is required for decryption) may be different from the algorithm itself.
- Feistel networks: block ciphers in which the input is divided into two halves, a nonlinear transformation is applied to the right half, the result is added into the left half, and subsequently left and right half are swapped.Some examples: DES, Khufu, CAST, and Twofish.

  An advantage of this approach is that the the same round function can be used for both encryption and decryption, while nonlinear function itself need not to be invertible. The output of one nonlinear function is input directly to the next one, which decreases the amount of parallelism but increases the propagation of local changes.

Other variants and extensions of uniform transformations and Feistel ciphers have been proposed (Snefru, RC5, MISTY2 and IDEA).

In order to decrease the complexity of implementation and increase the speed of computations, nonlinear transformations are applied only to small parts of the block data and linear transformations are used to spread local changes.

- Bit permutation (DES)

- Rotation (Khufu)
- Pseudo-Hadamard transformation (SAFER)
- Maximum distance separable (MDS) transformation (Twofish and Rijndael).

A maximum distance separable (*MDS*) transformation over a field is a linear mapping from a field elements to b field elements such that a composite vector of $a + b$ elements has the property that the maximum number of non-zero elements in any non-zero vector is at lest $b + 1$.

The distance (i.e. the number of elements that differ) between any two distinct vectors produced by MDS mapping is at least b + 1. It can be shown that no mapping can have a larger minimum distance between two distinct vectors, hence the term maximum distance separable.

4.1.3 Chaos theory and Cryptanalysis

The word *diffusion* has different meanings in different sciences.

Let $X, X1 \in \Xi$, $X \neq X1$, such that $\Delta X = X \oplus X1$. In other words, ΔX is the difference of the pair of plaintexts X and $X1$. Let $Y = F(X)$ and $Y1 = F(X1)$

- How far or how close $\Delta Y = Y \oplus Y1$ is to ΔX?
- If the value of $X1$ is fixed and we change X from the set Ξ, what are all possible values of ΔY?

Block encryption ciphers are designed based on a principle proposed by Shannon: nonlinear mappings are alternated with mixing functions.

Shannon wrote:

```
[...]In a good mixing transformation ...  functions are complicated,
involving all variables in a sensitive way.  A small variation of
any one (variable) changes (the outputs) considerably.[...]
```

What does *sensitive dependence on initial conditions* mean in cryptography?

- In chaos we first assume that the difference is small (infinitely small) and than we measure the effect of the mapping (dynamics) asymptotically, when time goes to infinity, with Lyapunov exponents.
- In cryptography we measure the effect of a single iteration of the transformation F on all possible values of the difference between two plaintexts.

4.1.3.1 Measure of diffusion

The most commonly used measure of diffusion in cryptography is called differential approximation probability (DP for short) and it is defined as

$$DP = max_{\Delta X \neq 0 \ \Delta Y} P(\Delta Y | \Delta X) \tag{4.5}$$

where

$$P(\Delta Y|\Delta X) = \frac{\# \{X \in \Xi | F(X) \oplus F(X \copyright \Delta X) = \Delta Y\}}{2^m} \tag{4.6}$$

Properties of DP:

- $2^{-m+1} \leq P(\Delta Y|\Delta X) \leq 1$.
- $P(\Delta Y|\Delta X) = 1$ means that there exists a difference ΔX which is always mapped with F to a difference ΔY.
- If $P(\Delta Y|\Delta X) = 2^{-m+1}$ then, for given $\Delta X, \Delta Y$, is uniformly distributed.
- DP is a measure of differential uniformity of the map F: it is the maximum probability of having output difference ΔY , when the input difference is ΔX.
- Decreasing the DP yields to increasing the complexity of the differential attack.

4.1.3.2 Measure of linearity

The most commonly used measure of nonlinearity in cryptography is called linear approximation probability (LP for short) and it is defined as

$$LP = max_{a,b \neq 0}(2P_{a,b} - 1)^2 \tag{4.7}$$

where

$$P_{a,b} = \frac{X \in \Xi | X \bullet a = F(X) \bullet b}{2^m} \tag{4.8}$$

In the last equation, $a, b \in \{1, 2, ldots, 2^m - 1\}$ and $\alpha \bullet \beta$ denotes the parity of bit-wise product of α and β.

Although one always assumes that F is a nonlinear mapping, it may happen that there exists a linear expression of F as an *equation* for a certain modulo two sum of input bits of X and output bits of Y :

$$X \bullet a \oplus F(X) \bullet b = 0 \tag{4.9}$$

If the expression is satisfied with probability much more or much less than 0.5, then F can be approximated with a linear mapping. On the other hand, if the expression is satisfied with probability close to 0.5, then F has *strong* nonlinear properties.

Properties of LP:

- Let $N = \# \{X \in \Xi | X \bullet a = F(X) \bullet b\}$. It easy to see that $0 \leq N \leq 2^m$.
- We write $A = a \bullet X$ and $B = b \bullet F(X)$. For $N = 2^m$, it follows $A \bullet B = 0$ is satisfied for all X, which means that there exists a linear expression for input and output bits. In a similar way, $N = 0$ means that $A \bullet B = 1$ is satisfied for all X. In both cases $LP = 1$.
- For $N = 2^{m-1}$, the expression $A©B = 0$ is satisfied with probability 0.5, and therefore $LP_{a,b} = 0$.
- The linear approximation probability is square of the maximal imbalance of the following event: the parity of the input bits selected by the mask a is equal to the parity of the output bits selected by the mask b.
- Decreasing the LP yields to increasing the complexity of the linear attack.

However, the systems used in chaos are defined on real numbers, while cryptography deals with mappings defined on finite number of integers (such as Galois fields). Nevertheless the discretization of a chaotic maps can lead to very strong block cipher. Parameter LP and DP allow a simple robustness evaluation against linear and differential attacks.

4.1.3.3 Differential Cryptanalysis

An i-round differential is a couple (α, β), where α is the difference of a pair of distinct plaintexts B_0 and B_0^* and β is a possible difference for the resulting i-th outputs B_i and B_i^*.

The probability of an i-round differential (α, β) is the conditional probability

$$P(\Delta B_i = \beta | \Delta B_0 = \alpha) \tag{4.10}$$

assuming that the plaintexts and the round subkeys are independent and uniformly distributed.

An m-round characteristic constitutes an $(m+1)$-tuple of difference patterns: $(\Delta B_0, \ldots, \Delta B_m)$. If $\Delta B_0 = \Delta B_m$ then the characteristic is called an m-round iterative characteristic; otherwise, is called an m-round differential trail.

The probability of a characteristic is the probability that an initial difference pattern ΔB_0 propagates to difference patterns $\Delta B_1, \ldots, \Delta B_m$.

Under the assumption that the propagation probability from ΔB_{i-1} to ΔB_i is independent of the propagation from ΔB_0 to ΔB_{i-1}, this probability is given by $P(\Delta B_1|\Delta B_0) \cdot P(\Delta B_2|\Delta B_1) \cdot \ldots \cdot P(\Delta B_m|\Delta B_{m-1})$ where $P(\Delta B_i|\Delta B_{i-1})$ is the probability that the difference pattern ΔB_{i-1} at the input of the round transformation gives rise to ΔB_i at its output.

The basic procedure of a differential attack on a r-round iterated cipher can be summarized as follows:

1. Find $(r-1)$-round differential (α, β) such that its probability is maximum, or nearly maximum.

2. Choose a plaintext B_0 uniformly at random and compute B_0^* so that the difference ΔB_0 is α. Submit B_0 and B_0^* for encryption under the actual key.

   From the resultant ciphertexts B_r and B_r^* , find every possible value (if any) of the last-round subkey z_r corresponding to the anticipated difference β. Add one to the count of the number of appearances of each such value of the last-round subkey.

3. Repeat step 1 and step 2 until some values of z_r are counted significantly more often than others.

   Take this most-often-counted subkey, or this small set of such subkeys, as the cryptanalyst's decision for the actual subkey z_r.

For the complexity (number of encryptions needed) of this attack holds:

$$Comp(r) \geq \frac{2}{p_{max} - \frac{1}{2^m - 1}} \tag{4.11}$$

where $p_{max} = max_\alpha max_\beta P(\Delta B_r = \beta|\Delta B_0 = \alpha)$, i.e., DP, and m is the block length.

4.1.3.4 Linear Cryptanalysis

An I/O sum $S^{(i)}$ for the i-th round is a modulo-two sum of a balanced binary-valued function f_i of the round input B_{i-1} and a balanced binary-valued function g_i of the round output B_i, that is,

$$S^{(i)} = f_i(B_{i-1}) \oplus g_i(B_i) \tag{4.12}$$

where a balanced binary-valued function is defined as a function that takes on the value 0 for exactly half of its arguments and the value 1 otherwise.

I/O sums for successive rounds are linked if the output function g_{i-1} coincides with the input function f_i of the next round. When ρ successive $S^{(i)}$ are linked, their sum,

$$S^{(1,\dots,\rho)} = g_0(B_0) \oplus g_\rho(B_\rho) \tag{4.13}$$

is called a multi-round I/O sum.

The imbalance $I(V)$ of a binary-valued variable V is the nonnegative real number $|2P[V = 0] - 1|$. The im balance is used as a measure for the *effectiveness* of an I/O sum. The average-key imbalance of the I/O sum $S^{(1,\dots,\rho)}$ is the expectation of the key dependent imbalances $I(S^{(1,\dots,\rho)}|z^{(1,\dots,\rho)})$ and is denoted as $I(S^{(1,\dots,\rho)})$.

An I/O sum is effective if it has a large average-key imbalance and is guaranteed if its average-key imbalance is 1.

Assuming that the attacker has access to N plaintext/ciphertext pairs with uniformly randomly chosen plaintexts the basic procedure is as follows.

1. Find an effective I/O sum $S^{(1,\dots,r-1)}$
2. Set up a counter $c[z_r]$ for each possible last-round key z_r and initialize all counters to zero.
3. Choose a plaintext pair (B_0, B_r).
4. For each possible value z_r, evaluate $(B_{r-1} = E_{z_r}^{-1}(Br)$ if $g_0(B_0) \oplus g_{r-1}(B_{r-1}) = 0$, increment $c[z_r]$ by 1.
5. Repeat Step 3 and 4 for all N available plaintext-ciphertext pairs.
6. Output all keys z_r that maximize $|c[z_r] - \frac{N}{2}|$ as candidates for the key actually used in the last round.

An $(m - 1)$-round linear expression can be turned into an m-round linear expression by appending a single-round linear expression such that all the intermediate bits cancel.

The linear approximation probability of the resulting linear expression is the product of linear approximation probabilities of the linear expressions involved in its construction.

The chain of m single-round linear expressions used to construct m-round linear expression is called m-round *linear trail.*

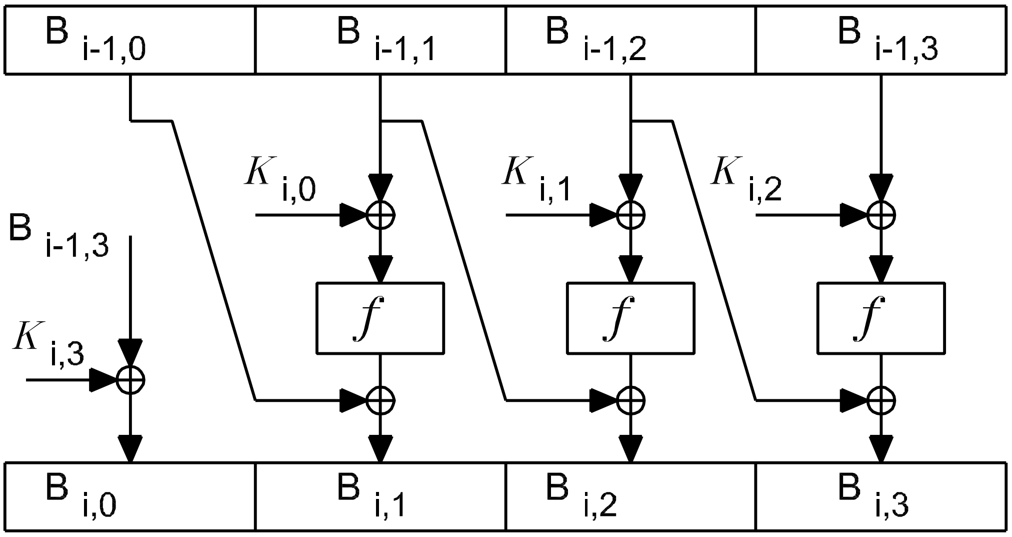


Fig. 4.1: Filippini system, round revisited ($n = 4$)

4.1.4 Filippini's encryption scheme

The well known cryptosystem DES [34] [35] is based on the Feistel Cipher and uses S-boxes (eight sample fixed tables, with 6 bits input and 4 bits output).

The system proposed by Filippini is an iterated block cipher, that differs from DES for two main aspects:

1. it uses an extended Feistel structure;

2. it uses dynamical systems instead of S-box.

Point 1) means that the system works by subdividing each single block in $n > 2$ subblocks, instead of the case $n = 2$ as used by the original Feistel Cipher. The length of a single block B_i of the i-th round is l_f bits; the system subdivides each block B_i in n subblocks $B_{i,j}$ where $0 \leq j \leq n-1$, which length is $l_f = l/n$ bits. Figure 4.1 shows the Filippini extended Feistel structure with $n = 4$ subblocks; the generic i-round transformation is obtained by:

$$\begin{array}{rcl} B_{i,0} & = & B_{i-1,n-1} \oplus K_{i,n-1} \\ B_{i,j} & = & B_{i-1,j-1} \oplus f(B_{i-1,j} \oplus K_{i,j-1}) \end{array} \tag{4.14}$$

$B_{i,j}$ is the j-th state byte in the i-th round and $K_{i,j}$ is the j-th byte in the i-th round key.

Point 2) means that Filippini's scheme considers a chaotic map M to generate the round function f. A similar approach has been followed also in [32] where the authors study block cipher based on functions obtained through a discretization of a non linear chaotic map. This work uses as round function $f : GF(2^8) \rightarrow GF(2^8)$ the results of a dynamic system with characteristics similar to chaotic ones:

$$x_i = f[x_{i-1}] = floor[256 \cdot [a^{(\frac{x_{i-1}}{255})} \mod 1]] \tag{4.15}$$

where $x_i \in GF(2^8)$ $\forall i$ and $a = 51$.

4.1.5 Linear and differential cryptanalysis: deterministic approach

LC and DC attacks are possible in a generic block cipher with x bits block length if there are, over all but a few rounds, predictable input-output correlations coefficients (difference propagations prop ratios) significantly larger than $2^{-x/2}$ (2^{1-x}) [33]. Prop ratio is the fraction of input pairs with a fixed difference that propagate into a fixed output difference.
Input-output correlation coefficient (difference propagation prop ratio) is the sum of the correlation coefficients of all linear trails (prop ratios of all differential trails) that have the specified initial and final patterns. Trails describe how a particular set of inputs propagates into the round sequence [33].

In [30], the scheme analysis emphasizes the single trail dominance in every difference propagation (or input-output correlation) with ratio (or coefficient) larger than previous higher threshold. Therefore, to be resistant against DC and LC, it is necessary that there are no trails with a predicted prop ratio (correlation coefficient) higher than 2^{1-x} $(2^{-x/2})$.

The main important aspect in the LC and DC investigation is the so called *activity pattern* which specifies the active functions positions in a trail. Active functions are characterized by nonzero bytes in the selection vectors (or difference of the states) at the transformation input. The pattern byte weight is the number of active bytes in it.

If the number of subblocks $n \gg 1$, the round transformation can be considered as an uniform mapping and, by following [33], correlation coefficient of a linear trail (prop ratio of a differential trail) can be approximated by the product of input-output correlation (prop ratios) of its active f-functions. Differential issue can be asserted only because scheme [30] is a Markov cipher.

Maximum input-output correlation coefficient and maximum prop ratio for f-functions are defined respectively LP and DP as explained in section 4.1.3.1 and 4.1.3.2. By defining j_i as the number of active f-functions in the i-th round it is possible to state that any correlation coefficient $Cp(\Xi)$ and any prop ratio $Rp(\Omega)$ of generic m-th round linear trail Ξ and differential trail Ω are upper bounded by

$$\begin{aligned} Cp(\Xi) &\leq (LP_f)^{j_1+\ldots+j_m}. \\ Rp(\Omega) &\leq (DP_f)^{j_1+\ldots+j_m}. \end{aligned}$$

that represent the worst system performance. Only the scheme structure determines the minimum trail active functions amount and only the chosen map determines LP and DP.

Regarding how active bytes propagate through a trail, note that in the worst case j_i is equal to 1 $\forall i$, so that $\sum_{i=1}^{m} j_i \geq m$. With r-round:

$$Cp(\Xi) \leq (LP_f)^r; \quad Rp(\Omega) \leq (DP_f)^r$$

For function (4.14) it is true that $DP_f = 2^{-4.6767}$ and $LP_f = 2^{-1.83}$. In this situation, if $r > 35$ and block with $x = 128$ bits is assumed, LC and DC attacks will not result to be advantageous. Note that result obtained with our deterministic approach is logically equivalent with the one proposed in [30], where only DC case is analyzed.

4.1.6 Improved chaos based system

By following the Wide Trail Strategy, a new round layer to avoid multiple-round trails with few active f-functions has been added, in order to increase the whole system diffusion. The addition of the new diffusion layer increases j_i, therefore producing a tight upper bound of $Cp(\Xi)$ and $Rp(\Omega)$, thus a more robust scheme.

The proposed scheme state has been organized like a matrix $4 \times N$, where $N > 4$ is the state column amount.

Exploration of active bytes propagation through the rounds has been realized in the following by using the same terminology of [36]. In a pattern A a column with at least one active byte is an *active* one. The *column weight* of A, $CW[A]$, is the amount of A active columns. The *byte weight of the j-th column* of A, $CW_j[A]$, is its number of active bytes. The *trail weight* (calculated as $\sum_{i=1}^{m} j_i$, where m is the round amount) is the sum of its round input activity pattern weights, $BW[A]$.

Figure 4.2(c) shows an example of activity pattern propagation. New scheme first layer, the ExtFeist previous scheme round, works on the row sequence of our byte matrix state. As diffusion layer chose the AES's one (see [36]), composed by the following linear transformations:

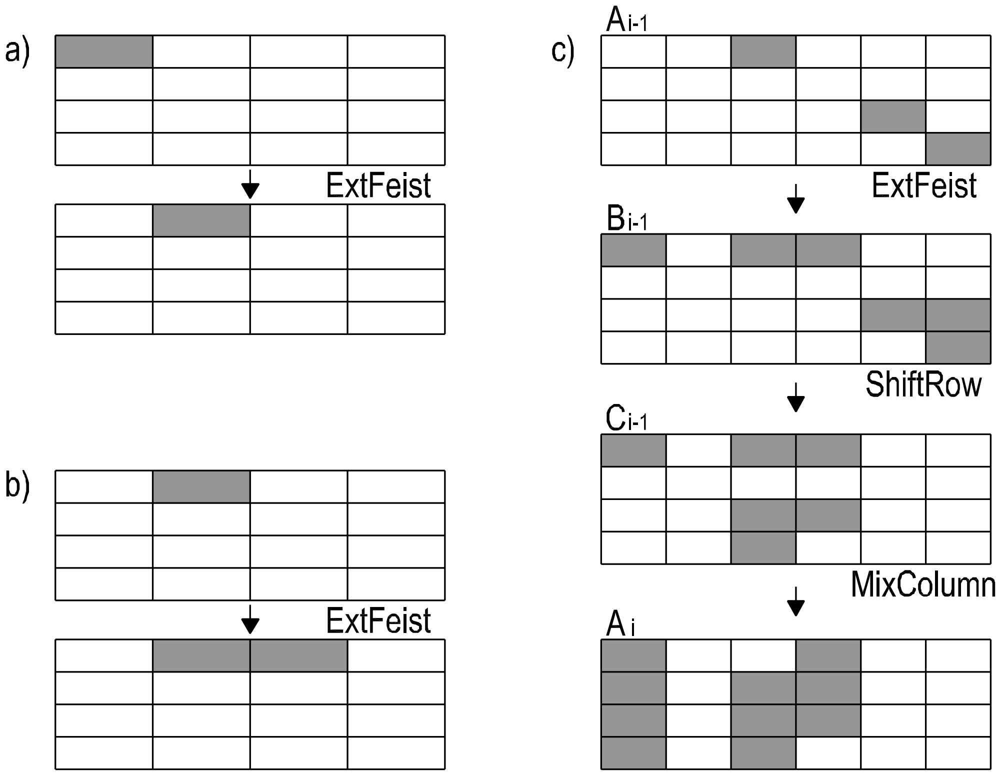


Fig. 4.2: Examples of activity pattern propagation into the round structure

- MixColumn - maps every bytes vector column of the input state into another vector column of the output state:
  - $CW_j[C_{i-1}] + CW_j[A_i] \geq 5$ (j is an active column);
  - $CW[C_{i-1}] = CW[A_i]$;
- ShiftRow - cyclically shifts every row of matrix state of a different number of position:
  - $BW[B_i] = BW[C_i]$;
  - $CW[C_i] \geq \max_j CW_j[B_i]$;
  - $CW[B_i] \geq \max_j CW_j[C_i]$.

The main difference with AES system consists in the introduction of the first layer, that can uses S-Box with non invertible functions, instead of the AES first layer which S-Box were based only on invertible functions. ExtFeist propagation effects are described using only differential ones by the following properties that emphasize the connection between the first layer input activity pattern and the output one:

- asymmetry in the scheme [30] produces only translation of active byte (4.2(a)). ExtFeist translates the active byte and propagates another and different one through a f-function (Figure 4.2(b)). The same result will be observed if the column weight are considered. More general situations may further be considered, but
$$BW[A_i] \leq BW[B_i] \leq 2 \cdot BW[A_i].$$
- Observe that adjacent bytes could influence each others: if right f-function output byte was equal to the left input byte, the right byte in pattern B could not be active. About the maximum column weight, in the worst case, all active bytes of $(j-1)$-column clear an equal number of active bytes from the j-column.
If the only A pattern active columns are the j-th and the (j-1)-th ones, where $CW_j[A_i] = k$ and $CW_{j-1}[A_i] = h$, then $max_j CW_j[B_i] = \max[k-h\ ,\ h] \geq \lceil k/2 \rceil$ if $k \geq h$.
Interactions within 3 or more columns cannot be excluded, but to make patterns which can combine right bytes over more than two columns require large pattern weight. The only way to have advantage from this situations is with $CW_j[A] = 4$, $CW_{j-1}[A] = 3$, $CW_{j-2}[A] = 2$, $CW_{j-3}[A] = 1$. In this situation, worst case propagation says $\max_j CW_j[B] = 1$. However, a large upper bound of this event probability is $DP_f^6 \simeq 2^{-28}$ and a reasonable conclusion can be that
$$\min_{B_i} \max_j CW_j[B_i] \geq \lceil (\max_j CW_j[A_i])/2 \rceil.$$

The above considerations come from the following theorems:

Theorem 1: The weight of two-round trail with Q active columns at the second round input is lower bounded by $3Q$.

Proof 1: If $CW[C_0] = Q$ then $BW[C_0] + BW[A_1] \geq 5Q$; therefore $BW[B_0] + BW[A_1] \geq 5Q$; worst bytes distribution is: $BW[B_0] = 4Q$ and $BW[A_1] = Q$; $BW[A_0] \geq \lceil \frac{BW[B_0]}{2} \rceil$, hence, if $BW[B_0] = 4Q$ then $BW[A_0] \geq 2Q$, therefore $BW[A_0] + BW[A_1] \geq 3Q$.

Theorem 2: In a two-round trail, the sum of the input and output active columns is at least 3.

Proof 2: G is a C_0 active column: $CW_G[C_0] + CW_G[A_1] \geq 5$. Clearly, $CW[B_0] \geq CW_G[C_0]$, thus $CW[A_0] \geq \lceil CW[B_0]/2 \rceil \geq \lceil CW_G[C_0]/2 \rceil$. We have that $\min_{B_1} \max_i CW_i[B_1] = \lceil \max_i CW_i[A_1]/2 \rceil$, therefore $CW[C_1] \geq \min_{B_1} \max_i CW_i[B_1]$. If $CW[C_1] = CW[A_2]$ it is possible to conclude that, independently from $CW_G[C_0]$ and $CW_G[A_1]$ value, $CW[A_0] + CW[A_2] \geq 3$.

Theorem 3: Any four rounds trail has at least 9 active bytes.

Proof 3: By following the same path of [36], the proof is completed by applying the first Theorem to the first and the last two rounds and the second Theorem to A_1 and A_3.

Concerning the r-round trails:

$$Cp(\Xi) \leq (LP_f)^{\frac{9}{4}\cdot r}; \;\; Rp(\Omega) \leq (DP_f)^{\frac{9}{4}\cdot r}$$

4.2 Conclusions

The presented analyses allows to conclude that, by considering the f-function (4.14) and $x = 128$, LC and DC attacks in the proposed modified system are not advantageous if $r > 16$ (while in the original Filippini's system we had $r > 35$). Simulations show that the time required to execute 16 modified system rounds is about the half of the time required to execute 35 Filippini's system rounds; moreover modified system decoding time is about 60% of the previous scheme required time (MixColumn map is faster than its inverse). Therefore by adding a diffusion layer, the cipher performances are greatly improved.

5. CONCLUSIONS

The work realized during the past three years and presented in this thesis was devoted to the research of new strategies to delivery multimedia content in both wired and wireless networks, able to guarantee a QoS management inside the network.

Thanks of all the mentioned studying directions, various strategies have been proposed and widely analyzed in this thesis, all of them showing very good performances and resulting in considerable enhancement of the networks QoS.

Chapter 2 has introduced the Peer-tree-Peer (P3P) platform, a peer-to-peer overlay network in which peer organize them self in order to obtain the best Qos results bounded by the imposed resource limitation and the underlying network properties.

P3P proposes a completely distributed approach to the problem of delivery multimedia content: a rendez-vous algorithm, continuous test of the underlying network performances, failure recovery, optimization procedure.

In order to obtain resilience against peer failure P3P also adopts a particular coding technique, Multiple Description Coding (MDC), that allow to realize overlay network path redundancy during the content delivery phase.

The behavior of P3P have been studied both by simulation that by emulation realizing the component software that provides all the P3P peer functionalities and testing the real behavior in dedicated laboratories. P3P can successfully realize the promised overlay network and minimize the resources required by each peer in order to provide the service to all the other one of the network.

Chapter 3 deeply analyzes the possibility of translate multicast paradigm into wireless sensor networks with critical energy consumption issues and extremely reduced computational resources.

This chapter proposes a new random geographic routing protocol, M-GeRaf, studied in order to minimize the amount of messages required to reach a set of destination.

Analytical model and simulation results lead to the definition of a new energy efficient delivery protocol that can be used to transmits multimedia data in sensor network with aggressive power-off strategies without the need of any signalation messages.

Chapter 3 also stresses the required precision in the position estimation related to the coverage area range defining the limit of M-Geraf paradigm application

Finally, the Wide Trail Strategy based method proposed in Chapter 4 is able to enhance the security offered by an encryption scheme based on an extended Feistel structure. This could be used to improve wireless networks security.

BIBLIOGRAPHY

[1] A. Odorizzi, G. Mazzini, ”A P2P Approach to Streaming Multimedia Contents in a E-Learning Oriented Platform,” IEEE IWWAN 2006, June 2006, New York.

[2] A. Odorizzi, G. Mazzini, ”P3P: a P2P Overlay Multicast Network for Multimedia Streaming,” NOLTA 2006, September 2006, Bologna, Italy.

[3] P. Francis,*Your own Internet distribution* Tech *rep at www.aciri.org/yoid*, UC Berkeley ACIRI Tech Report, Apr. 2000.

[4] Y. Chu, S.G. Rao, H. Zhang,*A case for end System Multicast* In Proc. ACM SIGCOMM, August 2001.

[5] Y. Chawathe,*Scattercast: an architecture for Internet broadcast distribution as an infrastructure service* PhD Thesis, U.C. Berkley, December 2000.

[6] J. Jannotti, D. K. Gifford, K.L. Johnson, M.F. Kaashoek,*Overcast: Relay multicsting with an overlay network* in *5th symposium on Operating System Designand implementation (OSDI)*, Dec. 2000.

[7] D. Pendarakis Tellium ,*ALMI: An Application Level Multicast Infrastucture* in Proc- of 3rd usenix symposium on Internet tchnologies and Systems, Mar. 2001.

[8] V.K. Goyal. *Multiple Description Coding: compression meets the Network*. IEEE Signal Processing Magazines, pages 74-93, September 2001.

[9] V.N. Padmanabhan, H.J. Wang, P.A. Chou,*Resilient Peer-to-Peer Streaming*. Technical Report MSR-TR-2003-11, Microsoft Research, Redmond, WA, March 2003.

[10] Yan Zhu, Min-You Wu, Wei Shu *Comparison Study and Evaluationof overlay multicast Networks.*

[11] D. Waitzman, C. Partridge, and S. Deering, *Distance Vector Multicast Routing Protocol* RFC 1075, November 1988.

[12] Krishna P. Gummadi, Stefan Saroiu and Steven D. Gribble, *King: Estimating Latency between Arbitrary Internet End Hosts* In Proc. SIGCOMM IMW 2002, November 2002, Marseille, France.

[13] *http://www.boost.org/*

[14] *http://www.ietf.org/rfc/rfc3550.txt*

[15] R.Jain, A.Puri, R.Sengupta. Geographical routing using partial information for wireless ad hoc networks. *IEEE Personal Communications MAgazine*, Feb 2001.

[16] B.Karp, H. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. *IEEE/ACM MobiCom*, Aug 2000.

[17] Y. Xu, J. Heidemann, D. Estrin. Geography-informed energy conservation for ad hoc routing. *IEEE/ACM MobiCom*, 2001, pp. 70-84.

[18] S.Wu, K.S. Candan. GPER: Geographic Rouitng in Sensor Netwroks. *12th International Conference on Network Protocols*, 2004, pp. 161-172.

[19] M. Zorzi and R.R. Rao. Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Multihop Performance. *IEEE Transactions on Mobile Computing* Vol. 2, No. 4. Oct.-Dec. 2003.

[20] L. Hu, D. Evans. Localization for mobile sensor networks. *MobiCom* 2004.

[21] P. Bergamo, G. Mazzini. Localization in sensor networks with fading and mobility. *PIMRC* 2002.

[22] E. L Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys. The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.

[23] Y.H.Lee, S.J.Horng. Fast parallel chessboard distance transform algorithms. *ICPADS* 1996, pp.488.

[24] R.Jain, A.Puri, R.Sengupta. *Geosubtended graphical routing using partial information for wireless ad hoc networks* IEEE Personal Communications Magazine, Feb 2001.

[25] S.Wu, K.S. Candan. *GPER: Geographic Routing in Sensor Networks.* 12th International Conference on Network Protocols, 2004, pp. 161-172.

[26] M. Zorzi and R.R. Rao. *Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Multihop Performance.* IEEE Transactions on Mobile Computing Vol. 2, No. 4. Oct.-Dec. 2003.

[27] A. Odorizzi, G. Mazzini, *M-Geraf: a Reliable Random Forwarding Geographic Routing Protocol in Multisink Ad Hoc and Sensor Networks.* IEEE ISPACS 2007, November 2007, Xiamen, China, pp. 553-556.

[28] A. Odorizzi, G. Mazzini, *M-GeRaf Analysis: Performance Improvement of a Multisink Ad Hoc and Sensor Network Geographical Random Routing Protocol.* IEEE SoftCom 2008, September 2008, Split-Dubrovnik, Croazia.

[29] A. Odorizzi, C.Taddia, G. Mazzini, "On the Application of Wide Trail Strategy to Improve an Extended Feistel Cryptosystem", IEEE Transactions on Circuits and Systems II, February 2006, vol. 53, num 2.

[30] A. Filippini, P. Bergamo, G.Mazzini, “Security Issues Based on Chaotic Systems”, IEEE Globecom 2002, pp. 148-152.

[31] L. Kocarev, *Chaos-Based cryptography: a brief overview*, IEEE Circuits and Systems Mag., Vol.1 N.3, pp. 7-21.

[32] G.Jakimoski, L.Kocarev, *Chaos and Cryptography: Block Encription Ciphers Based on Chaotic Maps*, IEEE transsactions on Circuits and Systems-I, Vol 48, No 2, Febryary 2001.

[33] J. Daemen, *Cipher and Hash Function Design Strategies Based on Linear and Differential Cryptanalysis*, Doctoral Dissertation, Chap. 5, March 1995, K.U.Leuven, http://www.esat.kuleuven.ac.be/∼cosicart/pub95.html

[34] M.J.B.Robshaw, *Block cipher*, RSA Laboratories Technical Report TR-601, August 1995.

[35] E.Biham, A.Shamir,*Differential cryptanalysis of DES-like cryptosystems*, Journal of Cryptology, Vol 4, No. 1, pp.3-72, 1991.

[36] J. Daemen and V. Rijmen, *Rijndael, the Advanced Encryption Standard*, Dr. Dobb’s Journal , Vol. 26, No. 3, March 2001, pp. 137–139.