Network Discovery in a Recursive Internet Network Architecture

S. Knappstein-Hamelink

Master of Science Thesis

TU Delft Delft University of Technology

Network Architectures and Services

PVM 2014-81

Network Discovery in a Recursive Internet Network Architecture

Master of Science Thesis

For the degree of Master of Science in Telecommunications at Delft University of Technology

S. Knappstein-Hamelink

9302287

26 February 2014

Thesis number: PVM 2014-81

Faculty of Electrical Engineering (EE) · Delft University of Technology

The work in this thesis was supported by Ministery of Defense - Royal Netherlands Navy. Their cooperation is hereby gratefully acknowledged.

Thesis number: PVM 2014-81

Committee Members: Dated: 26 February 2014

Supervisor(s):

prof.dr.ir. P.F.A. Van Mieghem

Reader(s):

dr.ir. F.A. Kuipers

dr. J.L.M Vrancken

C. Doerr

N.L.M. Van Adrichem, MSc

Abstract

The internet as we know it today was developed in the late '60 of last century as a robust communication system between different sites of the ministry of defense of the United States. In the following years other government sites and universities were connected as well and some years later, when the internet became an open standard, the internet became available for everybody around the world. Because of it success and the rush the internet took, problems which arose where put aside for the time being or quick fixes were made. A scientist in the United States, John Day, looked at the original requirements of the internet in his book "Patterns in Network Architecture: A Return to Fundamentals" and came up with a whole new theory called the recursive inter network architecture (RINA).

According to John Day, a better way to accommodate the current internet is to use independent recursive layers together with inter process communication (IPC) between these layers. An IPC process is a method of communication between different nodes in a network taking part in the same process. In a layer one or more distributed IPC processes (DIFs) can take place and a single node can take part in multiple DIFs. The layers are independent of each other, which has a number of advantages in security matters for example. In order to guarantee this independency, RINA uses logical addresses and point of attachment for single nodes.

In this thesis we have looked at finding a method to do network discovery through all layers and internally in the DIFs of the layers. First a node has to join the DIF in which it wants to communicate with another node. But before it can join that DIF it has to find that DIF. We found out that for finding the right DIF there is some sort of communication between the layers needed in order to do network discovery between these layers. We came up with two different strategies to find the right DIF; one which search for a common DIF by repeatedly joining the biggest up layered DIF till the common DIF is found and in the second strategy the nodes of the DIFs advertise which other DIFs could be reached via their DIF. Next to that, we looked at the complexity of the strategies and found out that the second strategy is less complex than the first one and therefor more efficient.

Secondly the shortest path towards the right DIF has to be found. In order to find the shortest path towards the right DIF, we made a proposal in the form of a pseudo code. In

this proposal all the possible paths from every node in a N layer DIF to the different nodes which also participates in the same N+1or N-1 layer DIF are calculated for every metric. The paths are compared to each other and the shortest paths are advertised to all the other DIFs the N layer DIF node is participating in.

Finally, when the right DIF is found and the shortest path towards this DIF is calculated, we have looked if current routing protocols could be used for routing in a DIF internally. The requirements for these routing protocols are that they should be scalable and able to handle multicast and multiple metrics. After comparing various existing routing protocols and algorithms, we came to the conclusion that only the optimized link state routing protocol (OLSR), the enhanced interior gateway routing protocol (EIGRP), the Babel protocol and the multicast adaptive multiple constraints routing algorithm (MAMCRA) meet these requirements.

At this moment a lot of hurdles still has to be taken before RINA can be used. One of the issues is that the current routers are not able to do routing between different layers at the moment. A solution to that is that the current routers have to be adjusted or that new routers have to be build. The most ideal situation is that these routers are able to route between the different layers and also within a layer and that the developed routing protocols are open standard. A problem with the current suitable routing protocols within a layer is that EIGRP is a Cisco developed protocol and is only used in Cisco routers. Other companies which develop routers can not use EIGRP. We think it would be better to develop a complete new open standard protocol in which the users of the network can put their requirements for that network. This protocol would be the same for every RINA network, but could be specified for every independent RINA network. In this way different QoS requirements could be realized.

Another issue is that RINA is still in concept. Various universities are doing research and are developing elements of RINA. For example, in January 2013, a project group consisting of the i2CAT foundation, Nextworks, iMinds and the Interoute was established for the IRATI project. The main goal of this project group is to bring RINA to an architecture reference model. This will be done by the design and implementation of a RINA prototype on top of ethernet. This project will take two years and is still in progress. Followers of RINA are gathered in the Pouzin society. Specific information, presentations, papers and video's are chaired in the RINA web page of this society.

Contents

Acknowledgments **xi**

1 Introduction **1**

1-1 Background . . . 1

1-2 Problem Description . . . 1

1-3 Research objective . . . 2

1-4 Research questions . . . 2

1-5 Overview . . . 3

2 Recursive internet architecture **5**

2-1 Today's internet . . . 5

2-1-1 A brief history . . . 5

2-1-2 Shortcomings of the internet . . . 6

2-1-2-1 Addressing problem . . . 6

2-1-2-2 Multi-homing . . . 6

2-1-2-3 Port allocation . . . 6

2-1-2-4 Security . . . 7

2-1-2-5 Network management . . . 7

2-1-2-6 Quality of service . . . 8

2-1-2-7 Summary . . . 8

2-2 Inter process communication layers . . . 8

2-2-1 Elements of a layer . . . 10

2-2-2 Processes taking place in a DIF . . . 12

2-3 Differences with peer-to-peer networks and private network-network interface . . . 14

3 Names and addresses within a recursive network **17**

3-1 Names and addresses . . . 17

3-2 Location dependent node address . . . 18

3-3 Multicast . . . 19

4 Routing Protocols **21**

4-1 Pro-active Protocols . . . 22

4-1-1 Link State Routing Protocol . . . 22

4-1-2 Distance Vector Protocol . . . 23

4-1-3 Wireless Routing Protocol . . . 24

4-1-4 Fish eye state protocol and zone routing protocol . . . 24

4-1-5 Source Tree Adaptive Routing Protocol . . . 25

4-1-6 Cluster-head gateway switch routing protocol . . . 25

4-1-7 Path vector protocol . . . 26

4-2 Reactive protocols . . . 26

4-2-1 Dynamic Source Routing Protocol . . . 26

4-2-2 Adhoc On Demand Distance Vector Routing Protocol . . . 27

4-2-3 Associativity-Based Routing Protocol . . . 27

4-2-4 Signal Stability-Based Adaptive Routing Protocol . . . 28

4-2-5 Other reactive protocols . . . 28

4-3 Multicast Adaptive Multiple Constraints Routing Algorithm . . . 28

4-4 Requirements for routing in RINA . . . 29

5 Network discovery in multiple layers **31**

5-1 Resource information base and resource exchange information protocol . . . 31

5-2 Network discovery and path selection in multiple DIFs . . . 32

5-2-1 Searching for the right DIF . . . 32

5-2-2 Network discovery within a DIF . . . 37

6 Pseudo code and the complexity of the different strategies **41**

6-1 Complexity in a single DIF . . . 41

6-2 Pseudo code and the complexity of the different network discovery strategies . . . 42

6-2-1 Strategy 1 . . . 43

6-2-1-1 Pseudo code of strategy 1 . . . 43

6-2-1-2 Complexity of strategy 1 44
6-2-2 Strategy 2 44
6-2-2-1 Pseudo code of strategy 2 44
6-2-2-2 Complexity of strategy 2 47

7 Conclusions, recommendations and future work **49**
7-1 Conclusions 49
7-2 Recommendations 51
7-3 Future work 51

A Delta-t versus TCP and UDP **53**
A-1 Delta-t 53
A-2 TCP and UDP 54

Glossary **61**

List of Figures

2-1 Overview of an IPC process [Grasa 2012] 9

2-2 IPC process in a layer. Adapted from [Day 2008] 11

2-3 Communication between two systems. Adapted from [Day 2008] 12

2-4 Communication between two applications using relay stations. Adapted from [Day 2008] 13

2-5 PNNI network 15

5-1 Example network 34

5-2 Layered presentation of the DIFs of example 1 35

5-3 Layered presentation of the DIFs of example 2 35

5-4 Layered presentation of the DIFs of example 3 36

5-5 Building the resource information base 38

5-6 Routing table with multiple metrics 39

List of Tables

2-1 Applications, addresses and point of attachments 10

4-1 Overview of different protocols and their ability to reach the RINA requirements . 29

Acknowledgments

I am grateful to the Royal Netherlands Navy, who gave me the opportunity to finish my master during working hours. And I would also thank the people from the NAS-group for the support they gave me during my master thesis. And a special thanks to Niels van Adrichem who gave me direct support and read and commented my draft versions over and over again, which gave food for thought and sometimes discussion. I would also thank my colleaque Ben van Asten, who has helped me so many times when I got stuck in the program LyX in which I wrote this thesis. Last I would like to thank my family and especially my husband Michel, who supported me for so many years when I was doing this study in the evening hours and during the weekends, having sometimes very little time for them.

Chapter 1

Introduction

1-1 Background

The current internet was originally developed as a robust communication system between different sites of the ministry of defense of the United States. Because of its success other government sites and universities were connected as well and when it became an open standard the internet grew exponentially. There were some problems, but quick fixes were made for some of the problems with the idea that the real problem would be fixed in the future. But because of the rush the internet took, a lot of these problems were never fixed. John Day [Day 2008], a United States scientist, looked at the original requirements of the internet and came up with a whole new theory called recursive inter network architecture (RINA).

1-2 Problem Description

According to John Day, a better way to accommodate the current internet is to use recursive layers together with inter process communication (IPC) between these layers. An IPC process is a method of communication between different nodes in a network taking part in the same process and will be further explained in chapter 2.

In a layer one or more distributed IPC processes (DIFs) take place. When a node wants to communicate with another node via a certain application/process it first has to find the right DIF. When the right DIF is found, the shortest path towards that DIF has to be found and the sending node has to join that DIF.

When the right DIF is found and the source node has joined that DIF, the routing within the DIF can take place to find the shortest path to the destination node. Because a node can take part in multiple DIFs, the shortest path towards the destination node is not always the route between the source node en destination node using only the physical links between the nodes in the DIF. It might be possible that a route from A to B via another DIF is shorter than the route in the original DIF.

But to find that shorter route via another DIF is not easy, because the DIFs are independent of each other. This means that the nodes from one DIF have no knowledge of which nodes are participating in other DIFs. In fact, they even don't know that these nodes exist. The only thing a single node knows is that it is part of multiple DIFs, each with its own address space. And if a node does not know which nodes in his DIF are also connected to other DIFs, it can never do any network discovery through the different layers. And when a nodes does not know that a node in his DIF also takes part in another common DIF, it can not compare the routes between the two nodes in the different DIFs. In other words a node is not able to find a shorter route via another DIF if it does not know that there is another route between itself and a node from its DIF in another DIF.

1-3 Research objective

The research objective for this thesis is:

To get a better understanding of the recursive layer theory of John Day and to find a working method to do network discovery in a recursive inter network architecture in particular.

Network discovery in a recursive network can be split into three parts.

1. Find the right DIF
2. Find the shortest path towards that DIF and join that DIF
3. Find the shortest path within that DIF towards the destination node.

1-4 Research questions

From the problem description the following research questions are extracted:

1. How does recursive networking work?
2. How do we find the right DIF?
3. What is the shortest path towards the right DIF?
4. How can network discovery be done in a recursive inter network architecture?
5. Are current routing protocols feasible for a RINA network?
6. What is the complexity of the different methods?

1-5 Overview

The overview of this thesis will be as follow. First the recursive internet architecture will be explained in chapter 2. In chapter 3 the different names and addresses RINA uses for the single nodes are explained more into detail, especially the location dependent node address. Next to that, we will explain how the use of the different naming can be used to perform multicast. In chapter 4 the most common routing algorithms used nowadays are discussed and we will look if these algorithms can be used in a RINA network. In other words, we look if the current routing algorithms meet the requirements of a RINA network. In chapter 5, we made a proposal for strategies of how we think the DIF the sender is looking for could be found in RINA network. In chapter 6, the pseudo code for those strategies is developed and the complexity of the network discovery strategies is analyzed. In chapter 7, we will finish with the conclusions, recommendations and future work.

Chapter 2

Recursive internet architecture

In this chapter a brief history about the development of the internet will be given and what the shortcomings of the current internet are. After that RINA will be explained more into detail and how RINA will solve some of the shortcomings of the current internet will be discussed.

2-1 Today's internet

2-1-1 A brief history

Internet, as we know it today, was designed in the late 1960's. The Department of Defense (DoD) of the United States started, as a reaction on the launching into space of the Russian Sputnik I, the Advanced Research Projects Agency (ARPA)[1]. One of the projects of ARPA was the development of a robust wartime digital communication system. At that time the DoD had only the telephone system for communication and no digital system at all. Several plans were proposed and in 1969 the ARPANET was born. The ARPANET was designed as a robust system, which means that in case of a partial destruction of the network by an attack or other error, the system was designed to re-route the data in a quick and safely manner with a minimum of data loss. In 1969, the University of California in Los Angeles (UCLA) was the first university that connected two computers with each other (one at UCLA and one at the Stanford university in Palo Alto) and by the end of 1969 already 4 hosts were connected to each other. In the following years the ARPANET became available for other universities, research organizations and government agencies for data exchange and email was developed in 1973. A year later in 1974 the transport protocol named transmission control protocol/internet protocol (TCP/IP) was developed, which made communication between nodes far more easier. More and more computer networks were connected to the ARPANET which eventually lead to the birth of the internet in the early eighties [Internetsociety].

[1]Because ARPA is an agency of the DoD it is also called Defense Advanced Research Projects Agency (DARPA)

2-1-2 Shortcomings of the internet

As mentioned above, the early internet was a research project that needed funds to operate. To get funds, results were needed. To get results in a short period, quick fixes were made to fix problems with the presumption that structural solution would be made later. But in a short period the internet became an enormous success and grew explosively. A lot of commercial activities started to use the internet and it became more difficult to implement changes. So the quick fixes became permanent fixes. Also some problems were forgotten over time as they ceased to exist or did not lead to problems in a short term. But recent history shows that some of the quick fixes are not sufficient anymore and need to be fixed in the near future. If solutions for these early problems could be found, the internet can be more efficient. In the following paragraphs some of the problems are addressed.

2-1-2-1 Addressing problem

When the ARPANET was developed no one expected that the ARPANET would out range the number of available addresses. At the beginning the address of an interface message processor (IMP) was only 8 bits long and the host addresses only 6 bits. Later in the late '70 this was expanded to 16 bits and later when we started to use IP-addresses it became 32 bits. In the early 90s it became clear that there were also not enough 32-bit IP addresses to accommodate all future users of the internet. IPv6 with 128 bit addresses should overcome this, but the number of addresses is not the real problem. The real problem is that the more addresses there are, the bigger the routing tables become. Networks are becoming unscalable because of these big routing tables.

2-1-2-2 Multi-homing

Besides the scarcity of available addresses, another problem is that there is no support for multi-homing in the current internet. This problem was adresses when Tinker air force base, Oklahoma, joined the net in the early seventies. They wanted two connections for a single host for means of robustness [Day 2008]. The problem is that the ARPANET sees this as two different addresses and thus as two different hosts. The used routing algorithm at that time did not know that those two hosts belong to one physical node. To solve this problem a logical address space and a physical address space is needed. But because not many users have this requirement, this problem has never been solved although many workarounds, such as the virtual router redundancy protocol (VRRP) [RFC 5798 2010], which creates virtual routers, for example, were found. The problem with this workaround is that it does not scale.

2-1-2-3 Port allocation

Another unsolved issue is the usage of fixed ports for applications. In the beginning of the internet there where just a few applications: a terminal protocol called telnet, a file transfer protocol (FTP) including some mail facilities and a remote job entry protocol (RJE). Because the developers then did not know how to build a protocol that assigns an application to a port for each session, they gave these applications a fixed port number for the time being. A port

number consists of 16 bits, which means that 65536 different port numbers can be assigned. These port numbers are subdivided into 3 categories [RFC 6335 2011], 0 - 1023 are the well known or system ports, 1024 - 49151 are the user or registered ports and 49151 - 65535 are the dynamic ports. In 2011 already 76% of the system ports and 9% of the user ports were already assigned. Because the number of developed applications is growing by the day, the need to dynamically assign a port number to an application is getting bigger and bigger.

2-1-2-4 Security

Also for security reasons it is not preferable to use fixed ports. Hackers could easily attack just one port of a host and block the whole service at that host. In any case security has become a big issue in the internet. Initially there was no security at all built in the internet. Only a limited number of machines were attached to the internet and everybody knew each other and trusted each other, hence security was not ought to be necessary. When the internet grew and members didn't know each other anymore, security became important. Authentication protocols, prevention against jamming, impersonation and eavesdropping were developed and encryption of data became common. Because each protocol took its own security measures, the overload and thus delay of data was doubled, tripled, etc.

2-1-2-5 Network management

In the '80s the number of private networks grew rapidly and interest grew in how to manage them. At that time ARPANET had a network management capability, but it was internal to the Raytheon company. The entire network could be observed and debugged from one management center in Cambridge and, as long as the interface message processor switch was not broken down, it worked very well [Day 2008]. It was primarily oriented towards controlling terminal networks, but at that time packet switching networks became more and more common. In packet switched networks events happen so fast that it is impossible for humans to intervene, so network management had to become an automatic process. The institute of electrical and electronics engineers (IEEE) as well as the internet engineering task force (IETF) started to work on it.

During the mid-'80 the common management information protocol (CMIP) [RFC 1189 1980] was introduced by the IEEE community. Somewhat later the IETF introduces the simple network management protocol (SNMP) [RFC 1157 1990] and high-level entity management system (HEMS) [RFC 1076 1988]. Although CMIP is event driven, object oriented and is capable to more actions than the SNMP, which using polling and can only set actions to alter the state of a management device, the internet adopted SNMP in the late 1980s [Day 2008]. Because of its simplicity, the common thought was that SNMP would need less coding then HEMS or CMIP, but it turned out that SNMP implementations are larger than either HEMS or CMIP. The problem with SNMP is that it has a very rudimentary structure and it is not object oriented. These two issues hamper the further development of the protocol.

2-1-2-6 Quality of service

ARPANET was designed as a neutral net in which all traffic needed to be treated equally. QoS requirements were not foreseen. Today, customers are willing to pay for services. This means that different classes of traffic have to be distinguished from each other to fulfill these QoS requirements.

2-1-2-7 Summary

In the last subsections the major shortcomings of today's internet are explained more into detail. To summarize the above subsections, the following issues need to be fixed when designing a new network architecture, like RINA:

1. Addressing should be done in a way that it is scalable
2. A separation between a logical and physical address space is needed to solve the multi-homing problem
3. A dynamic port allocation for each session is needed, because more and more applications are developed
4. A uniform security protocol instead of each protocol taking its own security measures is needed to overcome the extra overhead and extra delays
5. The network management protocol has to be reviewed as the SNMP is too rudimentary and hampers further development
6. QoS requirements should be taken into account

2-2 Inter process communication layers

To overcome the problems of today's internet several scientists tried to come up with a new structure of the internet. One of them is John Day which, with his book [Day 2008], he returned to the fundamentals and looked at the original requirements of the internet. He came up with a theory of using recursive layers that provide inter process communication (IPC). All layers have the same functions, but differ in range and scope.

In every layer an application process (AP) between two or more processing systems is running. This application performs and manages IPC and is called a distributed IPC facility (DIF). This DIF needs resources to exchange information with one or more processing systems, which it obtains from the layer below. That layer might not have all the resources by itself and obtains the missing resources from the layer below, where the process is repeated. In the end, the layer at which the physical media are, is reached. This is where the communication between two nodes takes place. In figure 2-1 DIF B, D, E and F are DIFs of the physical media. It is shown that the physical media is in RINA not just one physical layer like in the open system interconnection (OSI) [ISO/IEC 7498-1 1994] layer model, but the physical layer can be reached at different levels.

All layers are independent from each other. By using logical addresses instead of physical addresses the application of the Nth layer does not know what processes are taking place in the (N-1) layer. The application only knows its point of attachment, via a port-id with the (N-1) layer and trusts on the underlying layers to provide it the resources it needs for the communication. The nodes of an IPC process by itself have unique names within the DIFs they reside. These addresses are not visible outside the DIF.

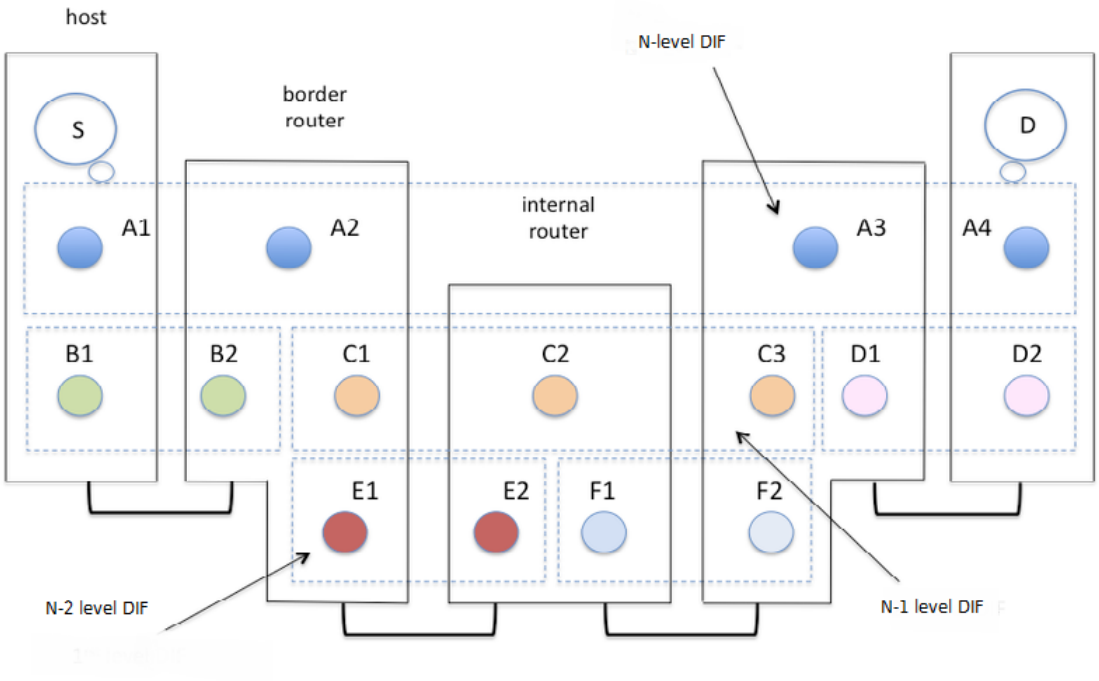


Figure 2-1: Overview of an IPC process [Grasa 2012]

If we look at figure 2-1 [Grasa 2012] we see that the source S is given the address A1 in the N layer DIF. A1 is in the same DIF as A4, which is the destination node D. A1 looks in its routing table if it has a route to A4 and finds out that the next hop to A4 is A2. A1 asks its N-1 layer if it can connect to A2 via its point of attachment. The N-1 layer sees the N-layer as an application. So the point of attachment to the N-1 layer and the name of a N layer application given by its N-1 layer is the same. Internally in a DIF, a node is given an address. In this example node A1 is given the address B1 in the N-1 layer. B1, in its turn, is in the same DIF as B2 which is in the N layer A2. In fact B1 and B2 represent the physical media between the nodes A1 and A2.

When the packet arrives at A2, A2 looks in its routing table if it has a path to A4 and sees that the next hop to A4 is A3. A2 asks the DIFs in its N-1 layer if one has a connection with A3. DIF C responds that it can connect with A3. In DIF C A2 is called C1 and A3 is called C3. The next hop to C3 from C1 is C2 and C1 asks his lower layer (N-2) DIFs if it has a

Level	Application name	Address	Point of Attachment name
N + 1	S		PA1
	D		PA4
N	PA1	A1	PB1
		A2	PB2, PC1
		A3	PC3, PD1
	PA4	A4	PD4
N-1	PB1	B1	Physical media
	PB2	B2	Physical media
	PC1	C1	PE1
		C2	PE2, PF1
	PC3	C3	PF2
	PD1	D1	Physical media
	PD2	D2	Physical media
N-2	PE1	E1	Physical media
	PE2	E2	Physical media
	PF1	F1	Physical media
	PF2	F2	Physical media

Table 2-1: Applications, addresses and point of attachments

connection with C2. The physical media connection between C1 and C2 is in the N-2 layer DIFs just like the physical media connection between C2 and C3, so the packet travels via E1 which is C1 in the N-1 layer and A2 in the N-layer to E2 which is C2 in the N-1 layer and from C2 the packet travels to C3 via F1 which is C2 in the N-1 layer and F2 which is C3 in the N-1 layer and A3 in the N-layer.

The last step is from A3 to A4. A3 has a direct link towards A4 via DIF D in the N-1 layer. Via D1 and D2 the packet arrives at A4, its destination.

In table 2-1 the process of figure 2-1 is organized in application, addresses and point of attachments names. The addresses are the location dependent names of a node while the application names and point of attachments names represent the location independent N+1 and N-1 layer. In table 2-1 the P stands for the point of attachment/application names. The different names in a RINA network will be explained further into detail in chapter 3.

2-2-1 Elements of a layer

A layer consists of different elements: application processes, application protocols, port ids, multiplexers, drivers and IPC processes like IPC management, inter process communication access protocol (IAP) and error and a flow control protocol (EFCP). The use of application processes, application protocols, port ids, multiplexers and drivers does not differ from the current use in today's internet, so the focus lies on the IPC process. In the IPC process mainly three sub processes are taking place: IPC transfer, IPC control and IPC management (figure 2-2).

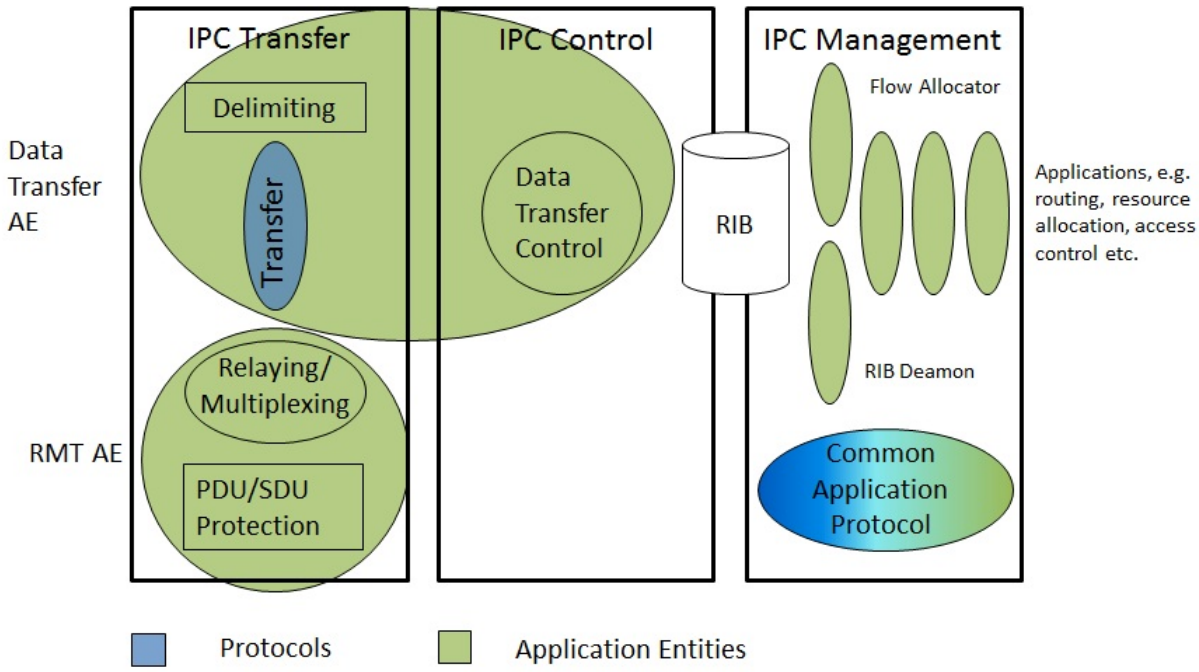


Figure 2-2: IPC process in a layer. Adapted from [Day 2008]

First we discus the IPC transfer functionality. The IPC transfer functionality actually forwards data across, just like the transmission control protocol (TCP) and the user data gram protocol (UDP) do in the OSI layer structure. The IPC functionality uses the delta-t protocol for this, while the current internet uses TCP/IP. In appendix A the difference between delta-t and TCP/IP is explained.

One of the functions of the IPC transfer functionality is delimiting and sequencing of protocol data units (PDUs) and distinguishing flows. Delimiting means that big data streams are fragmentized into smaller packets which can be send individually through the network. By giving these packets a sequence number (e.g. sequencing), the packets can be reconstructed in the right way at the receiver, as packets can arrive out of order. The IPC transfer functionality also executes PDU protection (both cyclic redundancy check (CRC) and/or encryption), multiplexing and relaying.

Next, the IPC control function handles the EFCP, IAP and quality of service (QoS) requirements. It sends retransmissions, acknowledgments and performs flow control. The IAP task is to find the destination address just like the (ARP) does in the internet, using the resource information exchange protocol (REIP), and it also performs access control. The IAP first checks if the policies and QoS requirements are met and then establishes an EFCP binding between the application protocol machine (APM) and the IPC process.

At last the functions of the IPC management are enrollment, routing, mapping N layer names to N-1 layer names and addresses, resource allocation and security management (authentication, protection, access control, etc). Between the IPC control and management task a resource information base (RIB) is maintained which is a logical store of local information of the state of the DIF. The RIB can be compared with a comprehensive routing table.

Summarized, there are only two protocols in a layer, being a data transfer protocol and a management protocol. Overall, there is also a common distributed application protocol (CDAP) and an authentication module. These two modules perform application connection establishment, authentication and operations like create/delete, read/write, start/stop, etc. They are used for maintaining the RIB, flow allocation and when joining or leaving a DIF. Next to the protocols we have five IPC modules: (1) delimiting, an error and flow control protocol consisting of the (2) data transfer and (3) data transfer control, (4) relaying and multiplexing and (5) PDU/SDU protection.

2-2-2 Processes taking place in a DIF

Two application processes, A and B, want to communicate with each other (see figure 2-3). A asks IPC to create a channel to application process B. First the IPC has to find application B [Grasa 2012]. The IPC management looks up the application name of B in the RIB and selects the appropriate DIF. In the RIB all names of the applications are stored and in which DIF the application is found. The RIB is updated by the resource information exchange protocol. When the destination application name is found, the RIB returns a local identifier of the appropriate interface of the destination.

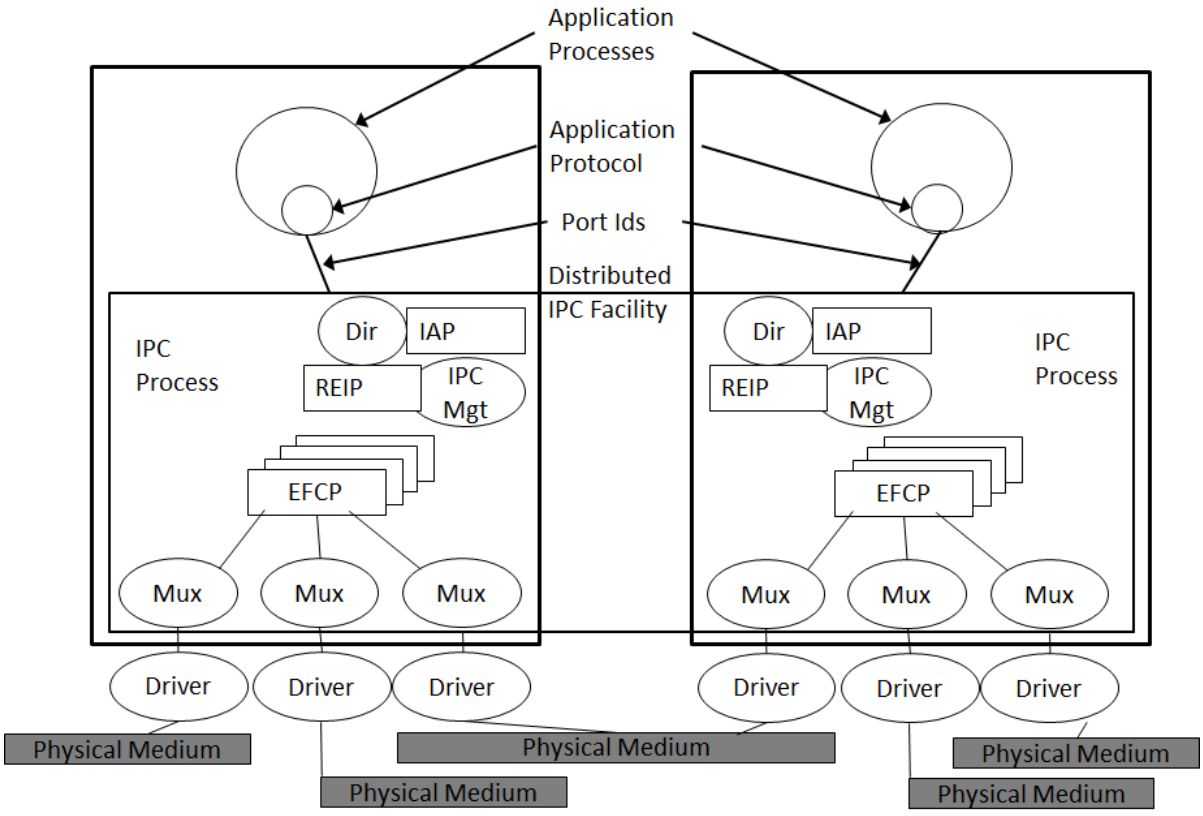


Figure 2-3: Communication between two systems. Adapted from [Day 2008]

If the application name is not found, the IPC access protocol is used to find B. In other words, the network discovery is started. IAP sends out a request for a connection to the IPC processes in application B. This request contains the destination name, source name, requested application process name, access control information, quality of service and termination condition. The IPC facility assigns a port-id ***a*** for communication with A. When the destination application B is found a error and flow control protocol connection is created and authentication takes place to see if the requesting application has access to the destination application. If positive, application B is notified of the IPC request from A and B is given a port-id *b*. This process is called enrollment. The sender is now given a name in the DIF (A1 in figure 2-1) by the DIF management and the destination too (A4 in figure 2-1).

After the enrollment the allocation of resources takes place through the different layers and data transfer can take place. Since multiple IPC processes can take place in one layer, which uses various media (for example A1 in figure 2-1 wants to send an email to A3 via a wired channel and an email to A4 via a wireless channel), multiplexing is used. For each IPC process a different EFCP connection is created and EFCP connections that uses the same medium are multiplexed before the data is sent over the physical medium. The connection identifier identifies to which multiplexer the EFCP connection belongs to. Multiple streams of processes can take place in parallel, sharing resources. When finished, the resources are released and de-allocated. The receiver distinguishes which data belongs to which IPC process using the connection identifiers to deliver the data.

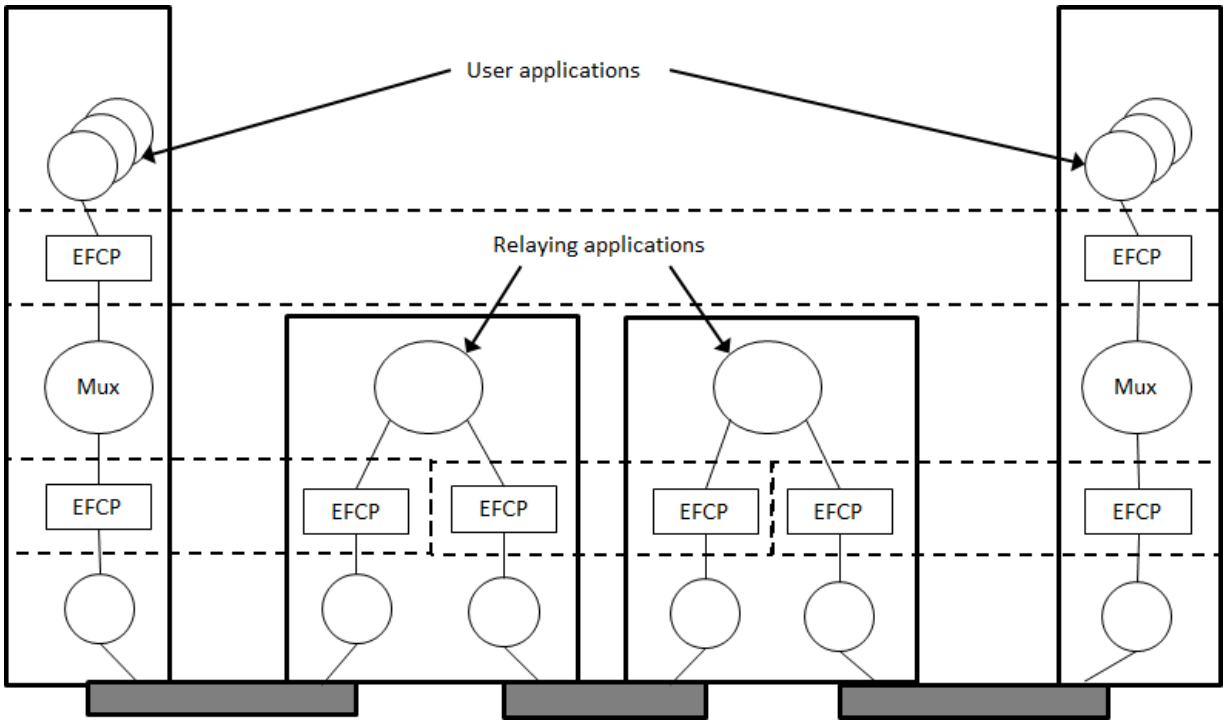


Figure 2-4: Communication between two applications using relay stations. Adapted from [Day 2008]

When communicating with N systems, relay systems can be used (see figure 2-4). A relay station has for each interface an own application protocol. If we use the IPC processes as names instead of the local names of the interfaces, multicast is possible. The relaying and multiplexing protocol (RaMP) interprets these names and send the data to the right IPC process.

The method described above is only valid when the source and destination are in the same DIF. When this is not the case the source needs to joint the destination's DIF up front or a whole new DIF has to be created. When the sender wants to join an existing DIF, the sender A requests via a common N-1 level DIF to establish an IPC channel with the destination B using B's AP name. The N-1 level DIF checks if B exists and if A has access to B. If so, authentication takes place. When the authentication is successful B assigns an N-level address to A. A establishes communication with all nearest neighbor members of the DIF and exchange REIP information with them. When a totally new DIF is created, the network management system (NMS) creates an IPC process and invites other members to join the DIF via its N-1 layer DIFs.

One of the questions that arise is how the IAP process actually does the network discovery and path calculation in the total network. It is all clear if network discovery and path calculation is done within the limits of one DIF and that the N-1 layer is only used as a physical connection between two nodes, but not when multiple DIFs in multiple layers are involved. In chapter 5 we will propose two different strategies of how a node in a N-layer can discover routes to a destination using one or more N-x level DIFs.

2-3 Differences with peer-to-peer networks and private network-network interface

Although a RINA network looks quite similar to peer-to-peer networks (P2P) and private network-network interface (PNNI), there are some mayor differences between these networks. A P2P network is an overlay network on top of an underlying topology. A node knows the identity of its neighbors and the identity of the destination. By sharing this information among neighbors, every node in a P2P network can make its own topology table [Androutsellis-Theotokis and Spinellis 2004].

At PNNI the network is divided into several hierarchical level, see figure 2-5. A group of nodes in a N-layer is represented by its peer group leader (PGL) in the N+1 layer. This PGL is chosen by election and summarizes the information from the members of its peer group before it distributes it among other PGLs at the N+1 level. This proces is also used the other way around, the data the PGL receives from other PGLs is distributed among the members of the peer group. All nodes in a peer group exchange topology state packets with each other. Different metrics are advertised in these topology state packets. By putting the PGL in the point of gravity of the peer group it can represent all different QoS parameters from the members of its peer group [Van Mieghem 2006]. The QoS measure distance from the point of gravity to all nodes is now more or less the same.

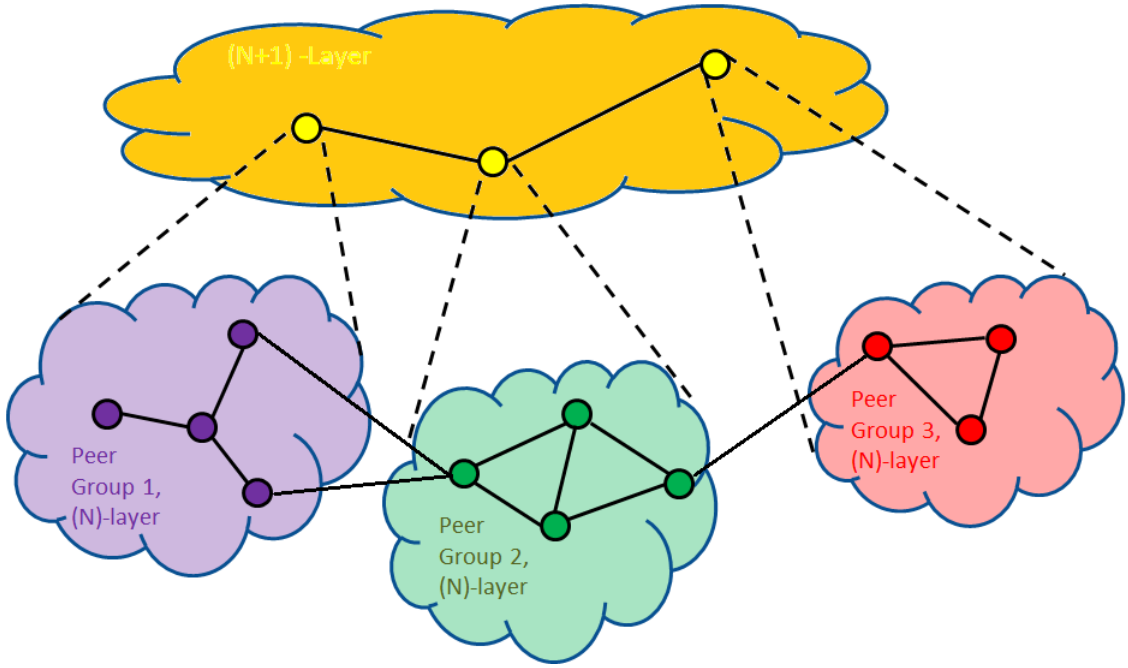


Figure 2-5: PNNI network

In RINA the users of an application (the members of a DIF) use the resources of their underlying DIFs to route data from one node to another, while in a P2P network the overlay network is used to route data to the underlying topology and in PNNI the PGLs. As in RINA the scale of the underlying layers decreases, the scale of a underlying layer in a P2P and PNNI network increases. But if we see a single DIF as a node, than there are more similarities with a P2P network. There is more research necessary to determine how far P2P and PNNI network technology can be used in a RINA network.

Chapter 3

Names and addresses within a recursive network

3-1 Names and addresses

To establish communication between two applications, the sending application needs to find the destination application. In order to find the destination application an IPC-process is used. According to Saltzer [Saltzer 1978] there are four necessary components of a network architecture: location-independent application names, location-dependent node addresses, point-of-attachment addresses and routes.

In a RINA network the N layer sees the N+1 layer as an application process (AP) and thus a DIF in the N+1 layer is given an application process name. The AP can migrate from host to host so the AP name has to be location independent [Day 2008]. To prevent double names, the names of the DIFs, the APs, are all taken from the same name space.

Within a DIF, every node has its own node address. These node addresses needs to be unique within a DIF, but instead of the names of the DIFs, a node address in one DIF can be re-used in another DIF. This does not give any problems because the DIFs function independent of each other and do not share addresses with each other.

To connect with lower layers the N layer DIF calls N-1 layer DIFs point of attachments (PoA). The definitions name, name space, address and address space according John Day [Day 2008] are listed on the next page.

Name
A name is a unique string, N, in some alphabet, A, that unambiguously denotes some object or denotes a statement in some language, L. The statements in L are constructed using the alphabet, A

Name space (NS)
A name space is a set {N} of names from which all names for a given collection of objects are taken. A name may be bound to one and only one object at time

Address
An address is a topologically significant name, which unambiguously identifies an object of a set of object

Address space (AS)
An address space is a name space defined over a set {A} of strings, a, in the language, L, which is a topological space

3-2 Location dependent node address

Within a DIF every node has a unique, location dependent node address. To maintain location dependent addresses, especially in a mobile environment where links are established and broken all the time, is not easy. If addresses are grouped together by location in the network, the address can give an indication in which direction the data has to be forwarded. Nodes near each other have similar addresses when grouped together and all routes to addresses within the same domain of granularity are the same. When an address within this domain has to be found the path to the destination is looked up in the routing tables at the entry node and the data is forwarded towards the destination. The addresses can still be route independent as there can be multiple routes towards the different addresses.

In RINA the layers have all the same function but have different scope. It depends on the scope whether all functions of the layer are used or not. Forwarding, for example, isn't always necessary in small scope networks. Layers segregate and aggregate traffic without the user noticing it. The scope of the lower layers is smaller that the one's on top of that layer, but the higher layers deal with less granularity over more elements. So the amount of work in all layers is more or less constant. When a hierarchical structure is used it is far more easier to manage the address space. Addresses within the certain granularity will get an address in which the first couple of numbers will be the same. This will specify the location of that address group (within a certain granularity) but not the route towards it. The more numbers are the same how smaller the granularity becomes and how further away from the root the address is.

3-3 Multicast

A group of addresses within a certain granularity can be called a subgroup of a network. The routes between the subgroups not always have to go upwards in the hierarchical structure and when a common subgroup has been found go downwards again. Shortcuts between subgroups are common. The advantages of these shortcuts are that the network is more robust and that the load of the network is more balanced. A disadvantage is that the shortcuts can cause loops in the network. A subgroup can also be used to overcome the multicast problem. The multicast problem consists of the fact that it is hard to send a message to a group of nodes in a network not using uni cast messages. A broadcast can be used, but then all the nodes in the network will receive the message, also the ones that do not need the data. This gives a lot of unnecessary overhead. For multicast a multicast application name is used. This multicast application name is allocated from a DIFs address space and names a set of addresses to which the applications are bound. For every multicast group a different multicast application name is used. Multiple subgroups can be formed, so scaling is not a problem anymore.

Chapter 4

Routing Protocols

Now that the difference between names and addresses is clear, the sending application needs to find a route to the destination application. There are different ways to find a route to a certain destination. Forwarding a packet from node to node and hope that it will finally reaches its destination is one option, but a more structured way of looking for a path will most probably lead to more success.

Normally network discovery is done by flooding. Flooding is a technique used to update topology databases [Van Mieghem 2006]. It is fast and robust method as long as the network is connected. When a node wants to discover a route in the network it sends a packet to it's neighbors. When the neighbors receive this packet they forward (floods) it to its neighbor and so on till the destination is reached or till all nodes are discovered. All intermediate nodes add their addresses to a list so that the destination knows which path to take to send back his acknowledgment to the sender. The sender now knows a path to the destination, but also the paths to the intermediate nodes. These paths are stored in the sender's routing table. A disadvantage of flooding is the big overhead. It is common that a node receives the initial request packet a multiple times via different routes. Different stop mechanisms are used to minimize this. Examples of stop mechanisms are the use of sequence numbers, a time to live (TTL) counter against bouncing, flood to all links except incoming link or selective flooding along a minimum spanning tree. Another disadvantage of flooding is, especially when the topology frequently changes, the wast of bandwidth due to an increasing number of collisions which are caused by sending a lot of packets through the network in a short time.

Next to network discovery there is path discovery. If a route towards a node is not in the nodes routing table already it sends out a request. This request is flooded through the network till it is received by the node it was asking for. This node returns with its address so that the sending nodes knows a route towards it and can store this route in its routing table. The calculation of the paths is done by a routing algorithm while the routing protocol disperse the information around the network and keeps the network state information up to date.

The network state information and thus the routing table is subject to changes. Nodes can join or leave the network and links come and go, especially when the nodes are mobile. The

the updating of the routing tables can be done by different routing protocols. Generally these protocols can be split in pro-active protocols and reactive protocols for inter domain routing and the border gateway protocol for exterior routing. In the following paragraphs the different routing protocol types including the used routing algorithms and the multicast adaptive multiple constraints routing (MAMCRA) routing algorithm are explained more into detail. After that the requirements for routing in RINA are pointed out and the different protocols and the algorithm are compared to the requirements. In the end a list of protocols and algorithm that suit the requirements best is left over.

4-1 Pro-active Protocols

Pro-active or table-driven protocols are protocols where every node maintains its own network topology information in routing tables. This can be done centralized or distributed, but in this thesis, because of the nature of RINA, where no central authority is used, only the distributed protocols are looked at.

4-1-1 Link State Routing Protocol

At link state routing, topology information is flooded through the graph, while the graph itself remains unchanged [Siva Ram Murthy and Manoj 2004]. The status of each link is sent over all the nodes and local routing algorithms compute the forwarding tables at the nodes. An example of link state routing (LSR) is the open shortest path first (OSPF) protocol [RFC 5340 2008, RFC 2328 1998] which uses Dijkstra's algorithm for computing the forwarding table [Van Mieghem 2006]. Dijkstra's algorithm is used when a network is unknown to a node. When a node want to send something to a certain destination it makes the best choice at each step in forwarding the packet and hopes it finds the best optimum at the end. Dijkstra's algorithm stops when the destination node is found or when all nodes are discovered.

Advantages of link state routing are that they are more scalable and are less bandwidth intensive than the distance vector protocol described in the next section. An other advantage is that the routing tables are quickly updated after a topology change so bouncing and the occurrence of loops is minimized. And if loops do occur they disappear when all the nodes have received the link-state update. The main disadvantage is that the routing tables are very big, in the order of n^2, which make the link state routing protocol memory-intensive. To overcome this OSPF uses hierarchical routing so that the number of nodes one node has to sent its updates to is limited and routing tables are not that big anymore.

A LSR protocol that also used hierarchical routing is the optimized link state routing (OLSR) protocol. OLSR is LSR optimized for the mobile domain. It makes use of multipoint relay nodes [RFC 3626 2003]. Multipoint relay nodes are representatives of a group of nodes. When a change happens or an update is required in a network a node sends the change towards the nodes in its group and to its multipoint relay node. The multipoint relay node distributes the change over the other multipoint relay nodes and so on. Also some recent research proved that multiple metrics can be used in OLSR [Laven and Hjartquist 2008, Alkahtani et al. 2006].

An other protocol that uses LSR is the intermediate system to intermediate system (IS-IS) protocol [RFC 1142 1990]. IS-IS uses authentication of LSR packets. If a router can not authenticate the packets because it has not the right authentication it can not be seen as a neighbor. In RINA the members of a DIF are authenticated in advance and also before data will be send, so IS-IS is not needed.

Another problem is when using multiple constrained paths, more and more updates of the network state information are needed. The value of these different metrics can vary a lot in time, because of the dynamic nature of the resources. Many QoS routing algorithms have been developed to find paths with multiple constraints [Kuipers et al. 2002] and a lot of proposals to reduce the number of updates needed in a QoS protocol are written [Kuipers 2006, Fu et al. 2008].

4-1-2 Distance Vector Protocol

A distance vector protocol (DVP) floods periodically a list of shortest paths towards every other node in the network and the first node of the shortest path towards this node [Van Mieghem 2006, Siva Ram Murthy and Manoj 2004]. Each routing table can update itself, if necessary, when a new shortest path to a node has been found. The routing tables increases linear to the number of nodes, n. These updates are sent periodically or when changes like adding/removing new nodes or links, finding better routes, etc, occur. Compared to link state routing the distance vector protocol is less complex and has less overhead.

A common used algorithm for finding the shortest path in the distance vector protocol is Bellman-Ford's algorithm. Bellman-Ford's algorithm is used when the network is known. It computes the shortest path by looking at the hop count off all the links and stops when all the links are explored. Because all routes towards every destination are stored in each node the route calculation process is very fast. An disadvantage of DVP is the high overhead it is creating when the nodes in the network are very mobile. The overhead is growing proportional with the number of nodes, so DVP does not scale very well in dynamic networks.

Examples of protocols that use the DVR are the interior gateway routing protocol (IGRP) [Siva Ram Murthy and Manoj 2004] and the routing information protocol [RFC 1058 1988, RFC 2453 1998] (RIP). RIP was one of the first route exploring protocols. It has a hop limit of 15, uses only the hop count as a metric and converge very slow. For this reasons RIP is not suitable for big networks. Another disadvantage is that there is no protection against loops in RIP.

To overcome some of the limitations of RIP, IGRP was developed. IGRP supports multiple metrics for each route and has a maximum hop count of 255. The different metrics are converted to one metric using a formula to compare different routes. A disadvantage of IGRP is that it does not use sub net masks. It assumes that all the sub network addresses within the same class have the same sub net mask, wasting a lot of address space.

One of the problems with the Bellman-Ford algorithm is that it doesn't prevent for loops, which can cause count-to-infinity problems. To overcome this problem loop-free distance vector protocols were developed. Examples of these loop-free distance vector protocols are

enhanced interior gateway routing protocol [Siva Ram Murthy and Manoj 2004] (EIGRP), destination sequenced distance vector protocol (DSDV) [Perkins and Bhagwat 1994] and the Babel protocol [RFC 6126 2011].

EIGRP stores three tables: a neighbor table, a topology table and a routing table. No periodic connectivity messages are exchanged between the nodes, only when a new node enters the network or when other changes occur route changes are send. EIGRP uses six different metrics (bandwidth, load, delay, reliability, MTU and hop count) to calculate his routes.

DSDV uses sequence numbers in the routing table to overcome loops. A disadvantage of this is when the topology of the network changes, new sequence numbers are necessary. So DSDV is not suitable for highly dynamic networks. Another disadvantage of DSDV is that only periodically updates are used. This can cause delays during the routing because packets has to wait in a node for an update of the network when their route has been broken for example.

The Babel protocol is quite new. It is a loop avoiding distance vector protocol used in IPv4 and IPv6 at the same time. It is designed for the mobile environment, but it works also in a wired network. It uses link quality measurements to find stable routes and able to use multiple metrics and multiple route metric computation strategies. It also can take radio frequency into account to avoid interference. Babel uses triggered updates and explicit requests for routing information, but also periodic hello messages. After a distribution, Babel re-converges quickly to a new configuration which is loop-free but not the optimal route. After that, Babel slowly converges to an optimal configuration using sequenced routes like in DSDV.

4-1-3 Wireless Routing Protocol

The wireless routing protocol (WRP) [Siva Ram Murthy and Manoj 2004] has a lot of similarities with the distance vector protocol. It uses the Bellman-Ford algorithm and maintains an up-to-date view of the entire network. The difference with DVP is that next to a topology table WRP also stores a distance table, a routing table, a link cost table and a message retransmission list. An advantage of maintaining all these tables is that broken links are found much quicker and route updates are disseminated faster, while less table updates are necessary. In other words WRP convergence much faster than DSDV and needs fewer table updates. On the other side larger memory and bigger processing power are needed because the complexity increases as well. This increases if the nodes are very mobile, so WRP is just like DSDV not very suitable for very large, dynamic, ad hoc wireless networks.

4-1-4 Fish eye state protocol and zone routing protocol

The fish eye state protocol (FSR) is a combination of link state routing and distance vector protocol. A node periodically exchanges information only with its nearest neighbors instead of flooding information through the whole network. Each node maintains a topology table of the network, a route table and a neighbor list like in LSR. FSR uses scope to decide the time interval of route updates. Scope is defined as the set of nodes that can be reached within h hops from the node [Siva Ram Murthy and Manoj 2004]. The smaller the scope, the higher the update rate. This results in a decrease of accuracy of the connectivity state of

the network when the distance increases. The goal is to reduce the total of update messages and thus the overhead. To identify the different updates sequence numbers are used. Because of the different update frequencies and the reduction of the overhead FSR scales very well. A disadvantage of FSR is that when the distance of the route increases the probability of a not accurate representation of the network within the node also increases which can lead to non optimal routes.

FSR look a bit similar like the zone routing protocol (ZRP) [Siva Ram Murthy and Manoj 2004]. In ZRP the nodes are grouped into zones. A zone has a certain radius and within that zone the distance between nodes is not greater than h hops. ZRP uses selective flooding between its nodes to do network discovery. The aim of using selective flooding was to lower the overhead but now the overhead only increases because zones are very much overlapping because there is no coordination among nodes, which result in nodes being part of multiple zones.

4-1-5 Source Tree Adaptive Routing Protocol

In source tree adaptive routing (STAR) [Siva Ram Murthy and Manoj 2004] protocol every node broadcasts its own source tree. This source tree is build up by the least overhead routing approach (LORA) which prefers paths with less overhead over the shortest path. Every node constructs a graph of the topology using the source-tree information broadcasted by its neighbors. Every node sends out update messages when changes in the network happen. This messages is rebroadcasted by every intermediate node till one node eventually has received the change in the network. The intermediate nodes in a route has to check for routing loops. It checks if the packet already has traveled through one of the nodes it has in its route towards the destination. If this is the case it discards the packet and it sends a route repair message towards the source of the packet containing the complete source tree of the intermediate node towards the destination. An advantage of this method is that it reduces the overhead, but the disadvantage is that most of the time the path is not optimal.

4-1-6 Cluster-head gateway switch routing protocol

The cluster-head gateway switch routing protocol (CGSR) [Siva Ram Murthy and Manoj 2004] uses a hierarchical structure. It organizes nodes into clusters. The nodes elect a cluster head which represent a cluster of nodes. The cluster head is chosen in a way that all the nodes in its cluster are only one hop away, which makes the cluster head the node with the highest connectivity of that cluster. By using for example different spreading codes between the clusters, bandwidth can be allocated more dynamical among the different clusters, improving reuse. Within the cluster the bandwidth is than shared via a token based scheduling system. Communication between clusters is done via gateways. This are nodes which belong to two or more different clusters.

CGSR is an extension of the DSDV protocol. Next to a routing table which contains the cluster head for every node in the network, each node maintains also a routing table which contains the next-hop node for every destination in the network. Routing towards a destination always goes via the cluster head and if needed a gateway node. It can be that this is not the shortest path. For example the gateway could be one of the neighbors of the source node,

but the source node has to route via the cluster head instead of sending the packet directly to the gateway. Another disadvantage is that at high mobility of the nodes, the node which is cluster head could change all the time, which makes the system unstable due to the multiple link breaks.

4-1-7 Path vector protocol

A path vector protocol maintains the path information of the entire path to avoid loops. The most commonly used path vector protocol is the border gateway protocol (BGP) [RFC 4272 2006]. BGP is the most used inter-domain routing protocol in the internet. A BGP router exchanges routing information with other BGP systems [Van Mieghem 2006]. Within the autonomous system (AS) a provider chooses its own routing protocol, but between the ASs BGP is used. Complete routes between a BGP router in an AS and another BGP router in another AS are stored in the RIB. The prefix of the IP-address defines the AS used.

BGP uses a distance vector protocol for routing and takes also network policies into account. Because BGP takes also network policies into account not always the shortest path is used. BGP is is loop-free because loops are recognized by the appearance of the same AS number in the path vector, but because BGP stores path vectors it also contains large routing tables. BGP is not only used as a inter-domain routing protocol. Within large ASs BGP is used internally because the interior routing protocol does not scale to that size. Also BGP is used for multihoming to get multiple access points of a single internet service provider.

4-2 Reactive protocols

Reactive or on-demand protocols are protocols that discover the route when it is asked for. When a node wants to have a route to a certain destination the route will be computed a-la-minute. In the next sub paragraphs different reactive protocols will be explained.

4-2-1 Dynamic Source Routing Protocol

Dynamic Source Routing (DSR) [RFC 4728 2007] floods a route request (RREQ) through the network if it wants to find a route. The destination eventually receives this RREQ and returns to the source with a route reply (RREP) packet, which follows the opposite route back to the source. This RREQ contains a sequence number and a TTL counter to avoid loops and transmission of the same RREQ on the same link. DSR allows piggy-backing of a data packet on the RREQ packet, which means that the data packet can be send along with the RREQ. The data packet contains the total path to its destination and when a link breaks or the delivery of the data packet to the destination fails in an other way, a RouteError (RERR) is generated. The source will now re-initiate the network discovery procedure.

The main difference with other reactive protocols is that DSR used no periodic updates, which reduces the overhead. The overhead is also increased by using caches at intermediate nodes. Intermediate nodes extract the information about routes when forwarding the packets and store that in their local cache. Another advantage is that DSR is also capable to use multiple

metrics [Asokan et al. 2008]. Disadvantages of DSR are that the query packets can be quite large because the route information is attached to the header of the query packet, broken links are not repaired locally and the connection setup delay can be quite high. Especially when the nodes are very mobile, the performance of the protocol drastically increases. DSR does not scale very well.

4-2-2 Adhoc On Demand Distance Vector Routing Protocol

The adhoc on demand distance vector (AODV) routing protocol [RFC 3561 2003] groups nodes into multicast groups and forms trees with these groups. The discovery process of AODV also starts with a route request packet broadcasted to the neighbors. This RREQ contains source, destination, from sequence number, to sequence number and the number of hops. All neighbors check if they have already a stored path to the destination, if not the RREQ is forwarded. When one of more intermediate nodes have a path to the destination and both paths arrive at a single node, the one with the most recent sequence number will send as a RREP back to the source. When a (intermediate) node can not forward the RREQ or reply the RREP a route error is sent.

The main difference with DSR is that AODV uses a destination sequence number to determine the most up-to-date path and in AODV not the whole path is included in the data packets [Siva Ram Murthy and Manoj 2004]. All intermediate nodes store valid routes to the destination and timers are used to delete these routes if no RREP is received. Advantages of AODV are that it is loop-free and scales easily to large numbers of mobile nodes [Ade and Tijare 2010]. Disadvantages of AODV are a big packet overhead due to multiple RREP to a single RREQ, bandwidth consumption by periodically send updates, high discovery delay and the fact that AODV uses only the hop count as a metric.

4-2-3 Associativity-Based Routing Protocol

Associativity based routing (ABR) selects the routes based on the stability of the wireless links [Siva Ram Murthy and Manoj 2004]. A link can be stable or unstable, based on the amount of periodic updates a node receives from its neighbors. If a route towards a destination is not available in the source route cache it floods a RREQ through the network. This RREQ contains the path it already has traveled and the update count for the corresponding node in the path. If one RREQ reaches the destination, the destination waits a time-interval to be able to receive multiple RREQ. The path with the maximum amount of stable links will be selected. If there are two or more paths with the same amount of stable links the shortest path will be chosen. This is actually one of the disadvantages of ABR, the shortest path is not always chosen as the best path. Another disadvantage is that when a link is broken the last node before the link was broken broadcasts a route repair packet to find a new route towards the destination. Repetitive broadcasts of this type can cause high delays during route repairs. The main advantage of this protocol is that it reduces path breaks because the most stable path is chosen. This results in less flooding and less overhead.

4-2-4 Signal Stability-Based Adaptive Routing Protocol

Signal stability-based adaptive routing protocol [Siva Ram Murthy and Manoj 2004] uses signal stability for finding routes. This protocol is, just like ABR, beacon based and measures the beacon's signal strength as a measurement of link stability. Every node maintains the beacon count and the signal strength of its neighbors. When a source wants to find a destination it sends a RREQ to its neighbors. When the RREQ travels over a weak link the receiving node discards the RREQ. When it is received over a strong link the node forwards the RREQ towards its neighbors and so on till the destination has been found. If a strong link breaks or becomes a weak link the end node of that link notify their corresponding end nodes of the path (mostly the source node and destination node). After such a route break notification packet the source node rebroadcasts the RREQ again in order to find a new strong path.

A disadvantage of the condition that a node that receives a RREQ over al weak link may not forward this anymore results in a lot route request failures, which causes another path-finding process which doesn't consider the stability criteria. This consumes a lot of bandwidth and it doubles at least the path set-up time. Also the most stable path isn't by definition the shortest path. An advantage is that more stable paths are found than any other adhoc routing protocol which reduces retransmissions.

4-2-5 Other reactive protocols

There are a couple of other reactive protocols which are all, for different reasons, not suitable for RINA. Location aided routing (LAR)[Siva Ram Murthy and Manoj 2004] and the flow-oriented routing protocol (FORP) [Siva Ram Murthy and Manoj 2004] use for example the geographical position of a nodes for finding a path, while RINA uses only location dependent addresses within a DIF and location independent names outside the DIF. The temporally ordered routing algorithm (TORA) [Siva Ram Murthy and Manoj 2004] is considered as unsuitable either, because TORA uses the ordering of the node names for network discovery and in RINA there is no ordering in the names of the nodes.

4-3 Multicast Adaptive Multiple Constraints Routing Algorithm

The multicast adaptive multiple constraints routing algorithm [Kuipers and Van Mieghem 2002] (MAMCRA) is a routing algorithm that calculates the multi-constrained optimal path. The MAMCRA algorithm makes use of the self-adapting multiple constraints routing algorithm (SAMCRA) which is based on the following four concepts [Van Mieghem and Kuipers 2004]:

1. Non-linear definition of the path length, $l_\infty(P) = \underset{1 \leq i \leq m}{max} \left[\frac{w_i(P)}{L_i}\right]$, where m is the number of metrics,$w_i(P)$ is the weight vector of the path of the i^{th} metric andL_i is the maximum value of the i^{th} metric.
2. k-Shortest paths algorithm, which is similar to Dijkstra's algorithm only the shortest path, the second shortest path, etcetera, up to the$k$-th shortest path, is stored together with the corresponding lengths.

3. Non-dominated paths, where path 2 should be dominated by path 1 if, for all metrics, $weight(path1) \leq weight(path2)$. When this is not the case, both paths must be stored in the queue.

4. Look-ahead concept, which proposes to compute the shortest path tree rooted at the destination to each node in the network for each of the m link weights separately. For each link weight component $1 \leq i \leq m$, the shortest value from the destination to a node is stored in the queue of that node. In total Dijkstra's shortest path algorithm is executed m times resulting in N-1 vectors with shortest values for each link weight component from a node n to the destination B.

MAMCRA has to calculate more paths because all shortest paths from the source to all the multicast members has to be calculated, before optimization can take place. A disadvantage of MAMCRA is that not always the most optimal path in terms of resource consumption.

Protocol	Scalable	Multiple metric	Multicast
OSPF	Y	N	Y
OLSR	Y	Y	Y
IS-IS	Y	N	Y
RIP	N	N	N
IGRP	N	Y	Y
EIGRP	Y	Y	Y
DSDV	N	Y	N
Babel	Y	Y	Y
WRP	N	Y	N
FSR	Y	N	N
STAR	Y	Y	Y
CGSR	Y	Y	Y
BGP	Y	N	Y
MAMCRA	Y	Y	Y
DSR	N	Y	N
AODV	N	N	Y
ABR	N	Y	N
SSA	Y	N	N

Table 4-1: Overview of different protocols and their ability to reach the RINA requirements

4-4 Requirements for routing in RINA

To overcome some of the problems of the current internet, RINA has requirements for routing. It must be scalable and multiple metrics should be taken into account. Next to that it must convergence quickly after changes, congestion control must be taken into account and multicast must be supported. An overview of the different protocols and the MAMCRA algorithm and their capabilities is given in table 4-1. The protocols that are scalable in

a dynamic, mobile environment, can do multicast and are able to calculate their shortest routes with multiple metrics are the OLSR, EIGRP, Babel, STAR, CGSR and the MAMCRA algorithm.

But CGBR uses elected cluster heads for all his routing, which could change a lot in a dynamic mobile environment. The change of cluster heads can make the network unstable, especially in a dynamic mobile environment. Next to that all the traffic goes via the cluster heads, making the protocol inefficient, because mostly non-optimal paths will be used instead of shortest paths. Also the STAR protocol uses most of the time sub optimal paths. For this reason these two protocols are less suitable for a RINA network. Remains the OLSR, EIGRP, Babel and the MAMCRA algorithm.

Chapter 5

Network discovery in multiple layers

In the earlier chapters we have seen that a node has at least 3 different names in a RINA network (application name, node address and point of attachment name), but could have much more names if it participates in multiple DIFs. For routing, routers need to know which names are synonyms of each other, so that they can calculate the most efficient routes between nodes using the different DIFs. In this chapter, we propose a way how routing information needs to be stored and disseminated within a RINA network and, next to that, we will make a proposal for an algorithm for network discovery in a RINA network using identifiers for the different names.

5-1 Resource information base and resource exchange information protocol

Within a recursive network, the resource information base (RIB) is, according to John Day, defined as the logical store of local information on the state of the DIF [Day 2008]. Each IPC process maintains a RIB. In other words, each IPC process has its own database in a single node, in a single DIF, which contains the routing table with the costs of different metrics like hop count, delay, etc. All the shortest paths between nodes are stored in the RIB and when new information arrives the paths are recalculated if necessary.

We propose, that when a node joins a DIF, it will send out a so called "hello" packet towards the other nodes in the DIF, just like in a regular non-vector network. In this "hello" packet the node ,tells the other nodes who it is and how it is called in other N-1 layer DIFs it is participating in. All the other nodes in the DIF can compare if they have a N-1 layer DIF in common and, if so, compare whether the path in the N-1 layer DIF between the nodes is shorter than the path in the N layer DIF or not. The shortest path will be stored in the respective RIB of the node.

The updating of the RIB is done by the resource exchange information protocol (REIP). The REIP is defined by John Day [Day 2008] as an application protocol internal to a DIF used

to exchange resource information among the IPC processes of a DIF. We propose that the REIP sends out messages about mapping of the node addresses in the N layer with the point of attachment name and thus application name of the N-1 layer. Furthermore it maintains the connectivity within the DIF and exchanges resource and allocation information between the members of the DIF. The information collected and distributed by the REIP is used for coordination within the IPC-processes of a DIF.

5-2 Network discovery and path selection in multiple DIFs

Routing in RINA has to be done in two dimensions. First, vertically through the layers, to find the right DIF. And when the right DIF is found, routing has to be done horizontally within the DIF to find the right node. Searching within a DIF is relatively easy, as current routing protocols can be used as long as they meet the RINA requirements. In chapter 4 we have showed that at the moment only four current existing routing protocols are suitable for a RINA network, being, OLSR, EIGRP, Babel protocol and the MAMCRA algorithm. But the difficulty is not in the horizontal discovery, but in the vertical discovery. How does a node find the right DIF in which the IPC process it wants to use runs and in which the node to which it wants to communicates is participating in? In the next paragraphs we propose how routing in and through the different DIFs should be done.

5-2-1 Searching for the right DIF

When a source node does not know where the destination node is, it first has to find in which DIF the destination node is situated. A reactive protocol is best suited for this, because the shortest paths towards a certain DIF can change frequently, especially when using multiple constraints. If we would use a pro-active protocol, it would mean that all possible paths to all DIFs should be recalculated frequently to keep the shortest path database up-to-date. This gives a huge overhead and the network size would not scale anymore. When using a reactive protocol, the route towards a certain DIF is calculated on demand. This means that there is only a query for the shortest path when a path towards a DIF is needed. For a network with recursive layers there is no such a protocol yet.

When a node joins the network, the node creates DIFs (physical connections) with its closest neighbors and starts to fill its own RIB. After creating the DIFs, the source node sends out a query to its neighbors to ask if they already have a route towards the DIF which contains the node it wants to communicate with. If one or more neighbors have a route towards the destination node, the source node asks the neighbor if it could join the DIF in which the destination node is participating.

When the source node has joined the corresponding DIF, first it will do a network discovery and build up his RIB. Then it looks for the shortest path towards its destination node and forwards its data towards the neighbor with the shortest path. If no neighbor has a route towards the destination node, it needs to find the DIF in which the destination node participates. We will look at two different strategies to find the DIF in which the destination node is participating in.

Strategy 1

In the first strategy, the source node asks its neighbors which is the biggest DIF they participate in and how large this DIF is. To find its largest DIF, a node looks in all the routing tables of the different up layered DIFs it participates in and selects the DIF with the largest number of nodes. The source node compares the size of the different largest DIFs of the nodes and selects the largest one. When the largest up layered DIF is found, the source node asks the corresponding neighbor if it could join this DIF. When joining the DIF with the largest amount of nodes, the bigger the chance, compared to joining a DIF with less nodes, that in that DIF a node already has a path towards its destination node. This process will repeat itself until the source node has joined a DIF in which a node has already a path towards the destination node.

During this process the source node fills its RIB with the shortest paths towards nodes in the different DIFs it joined. Within the DIF the source node participates in, this RIB is distributed and the node will also receive route updates of other nodes. In this way shortcuts and optimization of routes will be found. During the network discovery the node that just joined a DIF tells the other nodes of the DIF its name in the N-1 DIFs it participates in. The other nodes can then see if they have a DIF in common and if the path between the two nodes via that DIF is shorter or not. The shortest path is always stored in the node its RIB. Only the node itself knows which DIF it has to use to transfer data to a certain destination node.

Strategy 2

In the second strategy the nodes of a DIF advertise in their N-1 layer DIFs which other DIFs can be reached from their DIF. The nodes from a N layer advertise to their N-1 layers because a N layer DIF trust it's underlying DIFs. If not, the N layer DIF would not allow the node from the N-1 layer DIF to join the N layer DIF. When a node joins a N layer DIF it advertises to the other nodes which names it uses in its N-1 layered DIFs, but it also advertises which underlying DIFs it can reach. Summarized, the nodes of a DIF advertise which DIFs can be reached from their DIF, but not in what way or by which route that DIF can be reached. Because this happens in all layers, the nodes in the N layer DIF know exactly which N-1 layer and N+1 layer DIFs they can reach.

To make things more clear we give some examples. In the first example one of the neighboring nodes already has a route towards the destination node. At the second, third and forth example there is no path yet towards the destination node. In the second example the first strategy is used and at the third and forth example the second strategy. In the figure 5-1 an example network is given. In this example network, all the DIFs corresponding to physical media are left out for simplicity.

Example 1:

In figure 5-1 a network with three DIFs is shown. Assume that a new node Z wants to connect to node F. In this example node Z is situated close to node A and node B. To find the right DIF, node Z first creates DIFs with its closest neighbors node A and node B. Then it sends out a query to node A and node B to find out if they already have a path to node F. Node A participates in DIF I and node B participates in DIF I and DIF III. Node A and node B

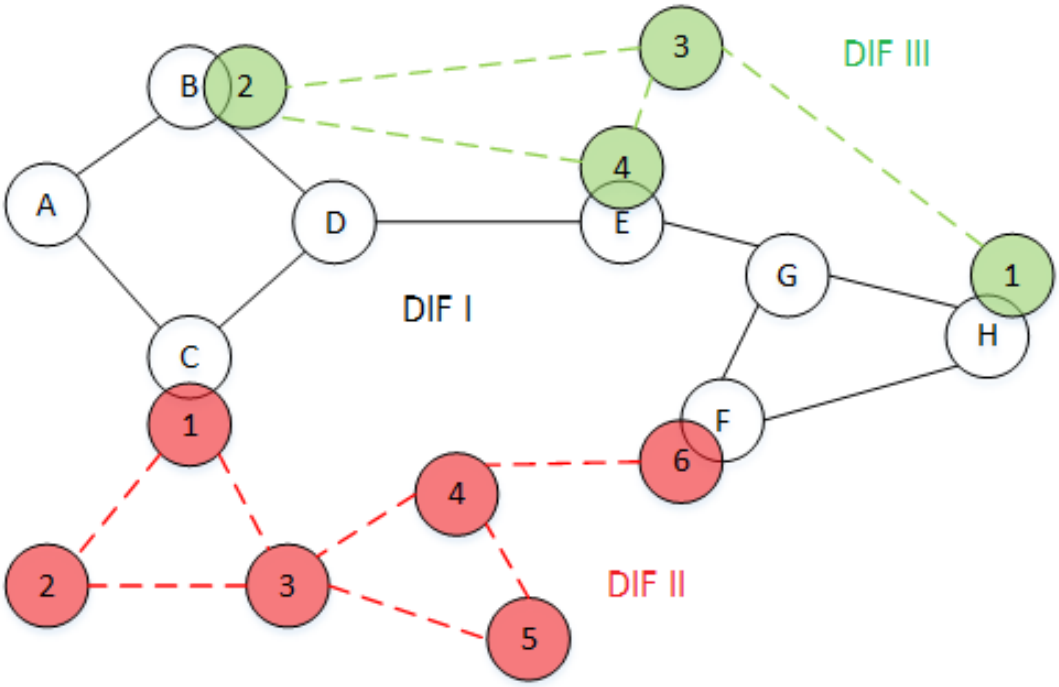


Figure 5-1: Example network

both have a path to node F in DIF I, respectively with hop count 5 and 4. But node B also has a shorter route in DIF III between node B and node E with hop count 1 instead of hop count 2 in DIF I. So in total, the hop count from node Z to node F, via node B, will be in total 4 via DIF III instead of 5 via DIF I only. The other nodes of DIF I do not know where node B routes the packets to, to get the shorter hop count, they only know that B has a route towards node E with hop count 1, so for them it looks like there is a new link in DIF I between node B and E. Node Z joins the DIF I and routes its packets to node F via node B. In figure 5-2 the layered presentation of this example, including some of the physical layers DIFs, is given.

Example 2:

Look at figure 5-1 again and now assume that node Z is situated close to node 3 of DIF III and node Z wants to connect to node F. Node Z creates a DIF with node 3 and asks if node 3 has path towards node F. This is not the case because node 3 does not take part in a DIF where node F also is a member of. Node 3 only participates in DIF III and in DIFs which represent the physical connection between two nodes. Because DIF III is the biggest DIF node 3 participates in, node Z asks node 3 if it could join DIF III. Now the process starts all over again when node Z sends out a query for node F within DIF III. Within DIF III, node 1, node 2 and node 4 all have a path to node F in DIF I with hop count 1, 4 and 2 respectively. Node Z joins DIF I and routes its packets to node F via node H. In figure 5-3 the layered presentation of this example, including some of the physical layer DIFs, is given.

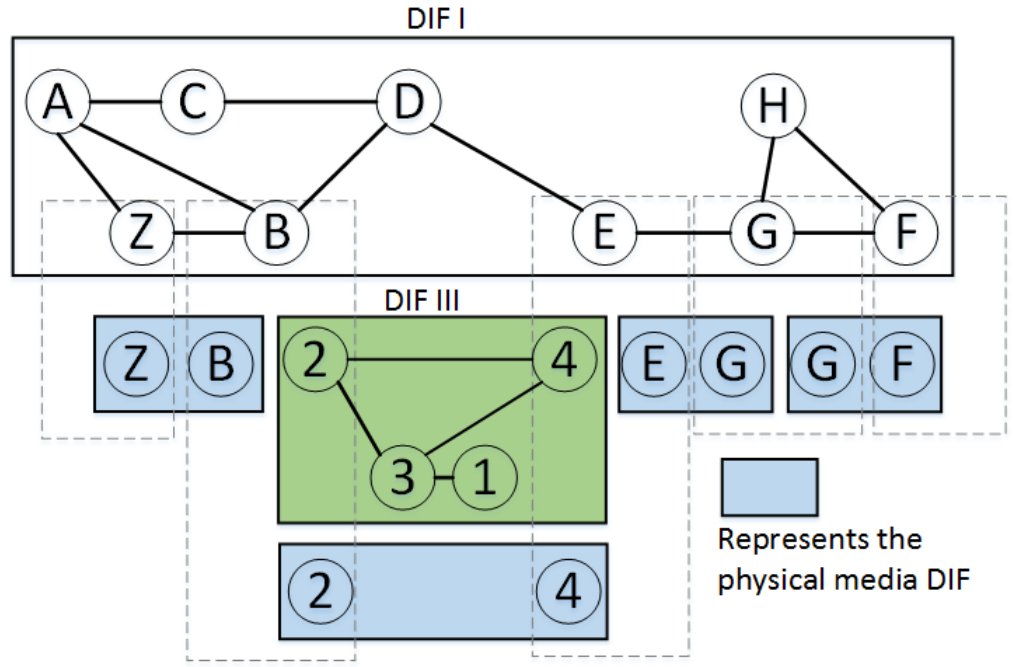


Figure 5-2: Layered presentation of the DIFs of example 1

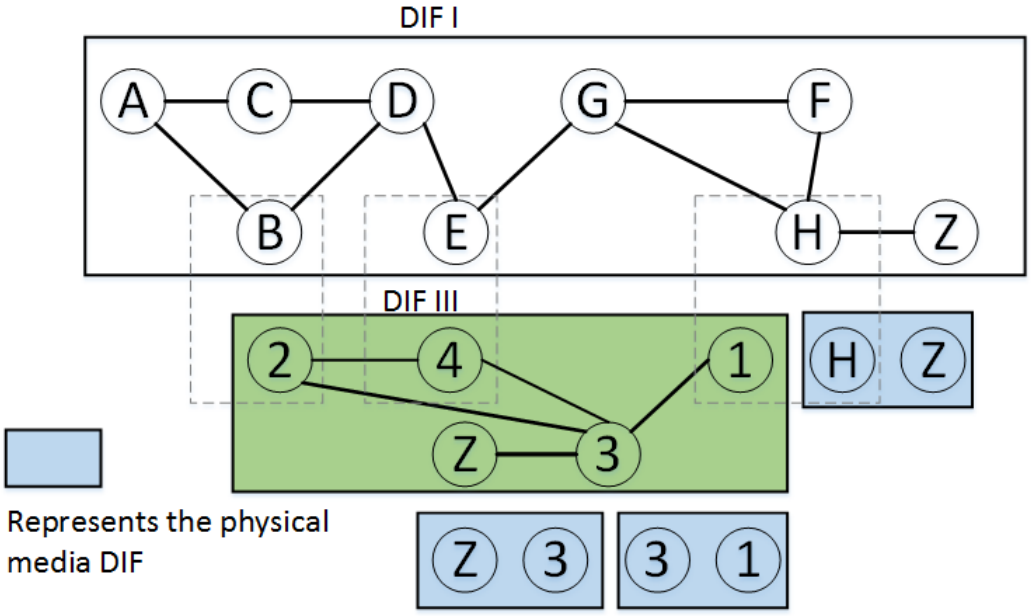


Figure 5-3: Layered presentation of the DIFs of example 2

Example 3:

Now take a whole new network. A node, called node 3 is connected with its neighbor node 1. Node 3 is also a member of a N+1 layer DIF, called DIF II (green in figure 5-4). Node 3 wants to connect to node X in DIF IV (red in figure 5-4). Node 3 knows that node X is in DIF IV, but it does not know how to reach that DIF. Node 3 asks its neighbor if it can reach DIF IV. Because all DIFs advertise to their lower layer DIFs to which DIFs they can forward, neighbor node 1 knows that it can reach DIF IV via DIF III. Node 3 joins DIF III and the same process happens. In DIF III node F can reach DIF IV, so a path to DIF IV is created.

If in example 3 strategy 1 instead of strategy 2 was used, node 3 would first have joined DIF I to find DIV IV instead of joining DIF III. Eventually it could (but not certainly) have found an upper layer DIF where node X was a member of, but that would not have been the shortest route towards node X as we see in figure 5-4.

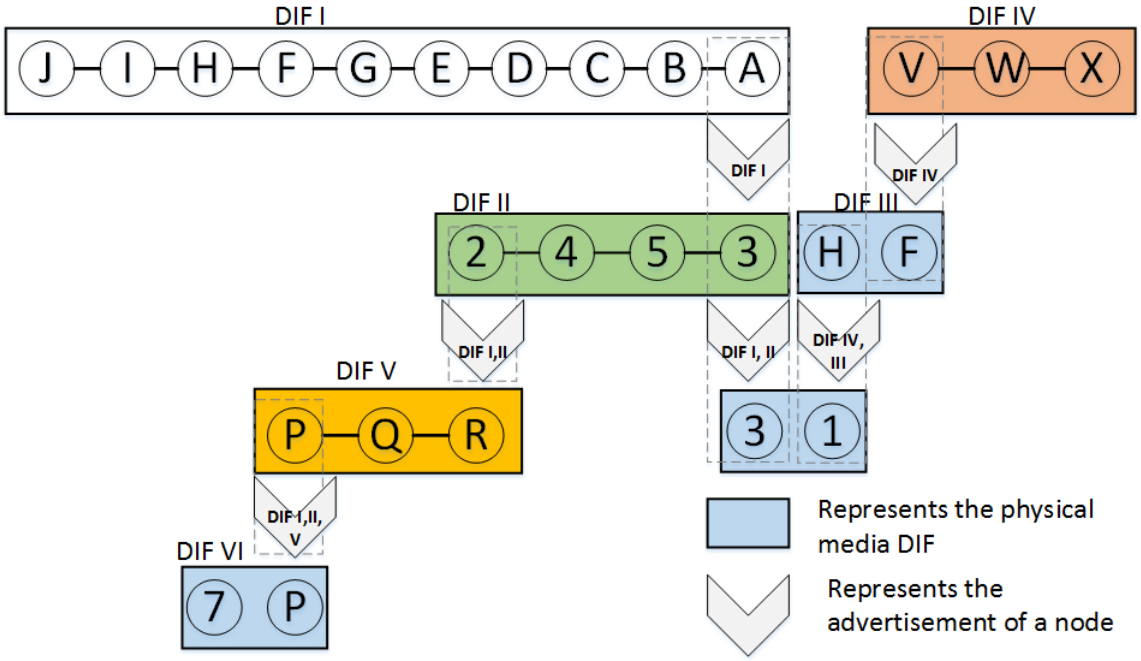


Figure 5-4: Layered presentation of the DIFs of example 3

Example 4

Take figure 5-4. Node 3 wants to connect with node 7 in DIF VI. Because node 2 advertises in DIF II that it can reach DIF VI, node 3 asks if node 2 can introduce him to the DIF that brings him closer to DIF VI. Node 2 introduces node 3 in DIF V and node 3 sends out a query for DIF VI in DIF V. Node P advertises in DIF V that it can reach DIF VI, so node 3 asks node P if it could join DIF VI. If positive, node 3 joins DIF VI and data can flow between node 3 and node 7.

Summarized, we have proposed two different strategies to find the right DIF. But what if the right DIF is not found? This means the DIF does not exist or the DIF do exist, but

we can not find it. In the first case a new DIF has to be created and the node we want to communicate with has to be invited to join that DIF. As we have seen in the paragraphs above, it is possible that the right DIF is never found when using the first strategy, but it is very rare to occur as we use strategy 2. If we use the breadth first search strategy or the depth first strategy [Cormen et al. 1991] to find the right DIF for example, all possible paths between the different DIFs are examined as we see a DIF as a node, thus the right DIF, if exist, will be found.

5-2-2 Network discovery within a DIF

As stated above, within a DIF every pro-active protocol can be used. It depends on the requirements of the DIF which one is the best suited. We have seen in chapter 4 that the routing protocol in a RINA network must be scalable, can handle multicast and multiple metrics. We have also seen that from the current existing routing protocols OLSR, EIGRP, Babel and the MAMCRA algorithm are the must suitable ones. In the next example is shown how network discovery within a DIF can be done.

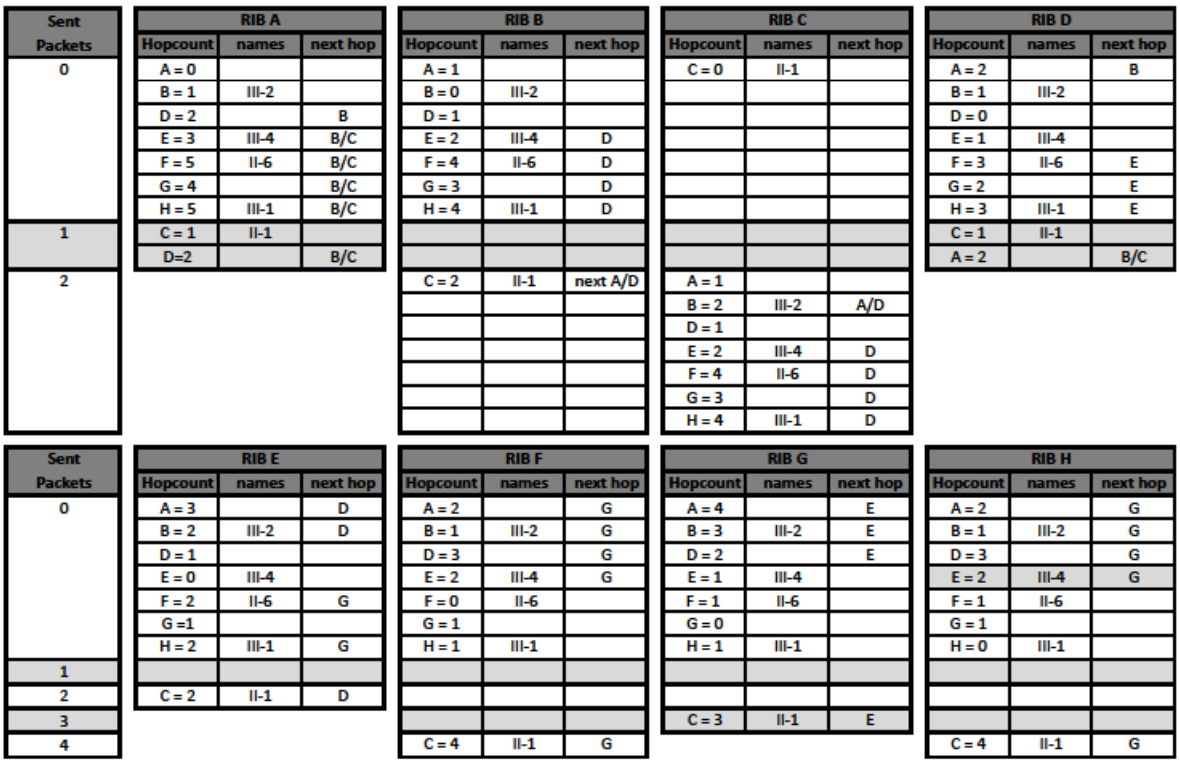
Example 5

Look again to figure 5-1 where a network with three DIFs is shown. DIF I is representing the N layer and DIF II and DIF III two N-1 layers. For the sake of simplicity the DIFs which represent the physical links between two nodes are left out of the figure. Assume that node C just joined DIF I and node C wants to do a network discovery. It sends a network discovery packet to its neighbors node A and node D in which it also tells that it also takes part in the N-1 layer DIF II with name 1 (II-1). Node A and node D will add their node name to the forwarding list of the packet and send the network discovery packet to their neighbors node B and node E. Node A and node D also send a return message to node C, which includes the RIBs of node A and node D. Node C now knows that node A and node D are one hop away and node C knows also all the latest routing information within the DIF.

If we look closer to the flooding of the network discovery packet of node C, we see that node B receives two of these discovery packets, one from node A and one from node D. It depends on which packet will arrive first at node B, to which node B will forward the discovery packet. To avoid loops, lower the overhead and to stop the flooding mechanism the discovery packet will never be sent back to a node that is already discovered and when a discovery packet is received at a node which already has received that discovery packet the flooding is automatically stopped. So when node B receives the discovery packet via node A before it receives it via node D, it will flood the discovery packet from node B to node D. If it then receives the discovery packet from node D, it notices that he already has forwarded a discovery packet from the link from node B, so does not send it again and the flooding stops partly at node B.

The flooding has not totally stopped yet because node D has also forwarded the discovery packet towards node E. Node E forwards the discovery packet to node G and so on, till all nodes are discovered. All intermediate nodes now has updated their RIBs.

During the discovery of node F, node F noticed that it has DIF II in common with node C. Node F compares the path towards node C in DIF I with the path in DIF II and finds out



Sent Packets
0
1
2

RIB A		
Hopcount	names	next hop
A = 0		
B = 1	III-2	
D = 2		B
E = 3	III-4	B/C
F = 5	II-6	B/C
G = 4		B/C
H = 5	III-1	B/C
C = 1	II-1	
D=2		B/C

RIB B		
Hopcount	names	next hop
A = 1		
B = 0	III-2	
D = 1		
E = 2	III-4	D
F = 4	II-6	D
G = 3		D
H = 4	III-1	D
C = 2	II-1	next A/D

RIB C		
Hopcount	names	next hop
C = 0	II-1	
A = 1		
B = 2	III-2	A/D
D = 1		
E = 2	III-4	D
F = 4	II-6	D
G = 3		D
H = 4	III-1	D

RIB D		
Hopcount	names	next hop
A = 2		B
B = 1	III-2	
D = 0		
E = 1	III-4	
F = 3	II-6	E
G = 2		E
H = 3	III-1	E
C = 1	II-1	
A = 2		B/C

Sent Packets
0
1
2
3
4

RIB E		
Hopcount	names	next hop
A = 3		D
B = 2	III-2	D
D = 1		
E = 0	III-4	
F = 2	II-6	G
G =1		
H = 2	III-1	G
C = 2	II-1	D

RIB F		
Hopcount	names	next hop
A = 2		G
B = 1	III-2	G
D = 3		G
E = 2	III-4	G
F = 0	II-6	
G = 1		
H = 1	III-1	
C = 4	II-1	G

RIB G		
Hopcount	names	next hop
A = 4		E
B = 3	III-2	E
D = 2		E
E = 1	III-4	
F = 1	II-6	
G = 0		
H = 1	III-1	
C = 3	II-1	E

RIB H		
Hopcount	names	next hop
A = 2		G
B = 1	III-2	G
D = 3		G
E = 2	III-4	G
F = 1	II-6	
G = 1		
H = 0	III-1	
C = 4	II-1	G

Figure 5-5: Building the resource information base

that the path in DIF II is shorter. Node F advertises in both DIFs the shortest path it has in DIF II with node C. Node F also advertises that it participates in DIF II with name 6 (II-6). Node C now knows that the shortest path to reach node F is via DIF II.

The reason that node F tells node C in DIF II that it also takes part in DIF II is because the nodes always trust their lower layered DIFs and the other nodes within DIF I does not need to know that there is another path between node C and node F in another DIF. They only need to know what the weight of that route is. During the network discovery of node C all other nodes can use the information they passed through to update their RIBs as well. And if something is changed in a RIB of one node it sends this change towards his neighbors, which can update their RIBs accordingly. In that way node C eventually knows the names of node B, E and H in DIF III as well. In figure 5-5 an overview is given how the RIB of the different nodes is build up.

As seen in the previous examples the network discovery of a node in a RINA network starts with creating DIFs to its neighbors and it repeats joining DIFs until the DIF it was looking for is found. Within the DIF a pro-active routing algorithm is used to find the shortest paths between the different nodes. For multiple metrics, every link in the network can be characterized by a m-dimensional link weight vector, where the component $m_i > 0$ is a QoS metric. The routing algorithm now calculates paths that obeys multiple constraints, $m_i(P) \leq L_i$ for all $1 \leq i \leq k$ [Van Mieghem and Kuipers 2004], where L_i is called the set of requirements of the user. There can be multiple paths that meet all the constraints but only the most optimal path will be stored in the RIB, see figure 5-6. When the constraints changes or the

$$\begin{bmatrix} 0 & \begin{pmatrix} m_1 \\ m_2 \\ . \\ . \\ m_k \end{pmatrix} & \cdots & \begin{pmatrix} m_1 \\ m_2 \\ . \\ . \\ m_k \end{pmatrix} \\ \begin{pmatrix} m_1 \\ m_2 \\ . \\ . \\ m_k \end{pmatrix} & 0 & \begin{matrix} . \\ . \\ . \\ . \end{matrix} & \begin{matrix} . \\ . \\ . \\ . \end{matrix} \\ \begin{matrix} . \\ . \\ . \\ . \end{matrix} & \begin{matrix} . \\ . \\ . \\ . \end{matrix} & 0 & \begin{pmatrix} m_1 \\ m_2 \\ . \\ . \\ m_k \end{pmatrix} \\ \begin{pmatrix} m_1 \\ m_2 \\ . \\ . \\ m_k \end{pmatrix} & \cdots & \begin{pmatrix} m_1 \\ m_2 \\ . \\ . \\ m_k \end{pmatrix} & 0 \end{bmatrix}$$

Figure 5-6: Routing table with multiple metrics

link metric changes the route calculation has to be redone and the shortest path can change significantly from the previous, because of the variety of possible routes.

Chapter 6

Pseudo code and the complexity of the different strategies

6-1 Complexity in a single DIF

We have seen that within every DIF, every routing protocol which suits the DIFs requirements can be used. This means that the complexity of that DIF in total has the complexity of the used routing protocol or algorithm. Before we look at the complexity of a single DIF, we first define the following parameters.

Definition 1
The complexity of a DIF is the complexity of the used protocol or algorithm in that DIF

Definition 2
Let N_{DIF} be the set of all nodes which take part in a N layer DIF and let $N_{DIF,max}$ be the number of nodes in the set N_{DIF}

Definition 3
Let N_{DIFUP} be the section of all nodes in a N layer DIF, which also take part in the same N+1 layer DIF

Definition 4
Let M be the set of metrics used in a DIF and let m_{max}be the maximum number of different metrics used in a DIF

Definition 5
Let L be the set of all links between the different nodes of the DIF

Definition 6
Let k be the number of shortest paths in a DIF

Definition 7
Let $w_m(i \rightarrow j)$ be the link vector between node i and node j which contains the weight between node i and node j per metric m

Definition 8
Let p be the number of multicast destinations in a DIF

If we take the pro-active routing algorithm MAMCRA, described in chapter 4, as an example, we see that the complexity of that algorithm is $O(kN_{DIF}log(kN_{DIF}) + k^2m_{max}L)$, where $k = k_{max} = O(exp(N_{DIF}ln(N_{DIF}))$ for the first part of the algorithm and $k = O(Np^2)$ for the second part [Kuipers and Van Mieghem 2002].

6-2 Pseudo code and the complexity of the different network discovery strategies

When a node joins the network and none of its closest neighbors already has a path towards its destination, the node first has to find the right DIF in which the destination node is situated. We have discussed two different strategies to to find the right DIF, one when the node joins repeatedly the biggest N+1 DIF till it has found a DIF which the destination node participates in, and one when the nodes in up layered and lower layer DIFs advertise to N layer DIF which DIFs can be reached from their DIFs.

Theorem
A recursive network is always less complex as a normal flat network

Proof

The total amount of nodes in a DIF is always less then the total number of nodes of the flat network because a DIF contains a subsection of the total amount of nodes. Also all calculations in the different DIFs can be done in parallel of each other. This automatically means that a recursive network is always less complex as a normal flat network, as long as the complexity of the strategies of finding the right DIF plus the complexity of the largest DIF in the network is less than the complexity of the flat network. ¤

6-2-1 Strategy 1

6-2-1-1 Pseudo code of strategy 1

For the first strategy we need to know the largest up layered DIF. To determine this, all nodes from a DIF have to determine the size of their up layered DIFs. This is done by comparing the number of nodes in each RIB of the DIFs with each other. When the largest up layered DIF of a node is found, it advertises the size of that DIF towards the other nodes in the DIF. These values are the input for the routing algorithm 1. The source node compares the different sizes of the DIFs and determines which up layered DIF is the biggest. When the largest up layered DIF is found, the source node joins this DIF and sends out a query if the destination node is participating in that DIF. If so, the routing algorithm used in that particular DIF can be used to route information from the source node to the destination node. If not, the process of finding the largest up layered DIF repeats itself.

Before the pseudo code is given in algorithm 1, first some extra definitions are defined.

Definition 9
Let UP_k be the set of N+1 layer DIFs in which node k participates in, and let $UP_{k,max}$ be the total number of up layered DIFs in which node k participates in

Definition 10
Let DIF_{max}be the largest DIF of the set of DIFs in which a node participates

Algorithm 1 Pseudo code strategy 1

1. DIF_{max} = NULL
2. for all nodes n ϵ N_{DIF}
3. for all elements of UP_k
4. if $N(UP_k)_{DIF,max}$ > DIF_{max}
5. DIF_{max} = $N(UP_k)_{DIF,max}$
6. end
7. end
8. join N+1 DIF (DIF_{max})

Proof

When looking at the pseudo code in algorithm 1 we see that in line 3 till 6 the largest up layered DIF for node k is calculated. This loop is repeated for all nodes in the original DIF except for the source node (line 2 till 7). In line 4 the size of the current up layered DIF is compared with the most recently found largest DIF. If the current DIF is larger, the number of the largest DIF is updated ¤

6-2-1-2 Complexity of strategy 1

Every nodes has to compare $UP_{k,max}$ times if that up layered DIF is the largest one it is participating in. The maximum number of uplayered DIFs is worst case $N-2$ which gives a complexity of $O(N)$. And every node in the DIF has to sent an update of its largest up layered DIF to all other nodes in the DIF, which gives a complexity of $O(N^2_{DIF,max})$. This operation has to be done multiple times, till the right DIF is found. In worst case this means that $N-2$ layers has to be crossed. The overall complexity of strategy 1 is therefore $O(N^2 N^2_{DIF,max})$. Algortihms with complexity bigger that $O(N^3)$ do not scale anymore, therefore strategy 1 is not suitable.

6-2-2 Strategy 2

6-2-2-1 Pseudo code of strategy 2

If we look at the second strategy, all nodes in a DIF advertise which DIFs can be reached via their DIF. In this advertisement also the weight to get to a certain DIF is advertised. These weight advertisements are the input vectors for the algorithm of strategy 2. When a DIF receives multiple options to reach a certain DIF it has to decide which DIF it will use. The shortest path towards a certain DIF has to be calculated.

When a N layer DIF and another layered DIF have more than one node in common all the possible paths from the nodes of the N layer DIF towards the nodes which are also participating in the other layered DIF has to be calculated. The links between the nodes in an N layer DIF have different weights, thus a route via node A towards another layered DIF has most certainly another weight than the route via node B towards that DIF. So when multiple nodes of one N layer DIF are also part of the same N+1 or N-1 DIF the N layer DIF advertises the weight using the shortest path per metrics in its DIF. This means that the shortest path from one metric can go via a different node than the shortest path of another metric. For the other layers this is not visible. They only know that a certain DIF can be reached via that DIF with a certain cost and how the information is routed towards that DIF is not visible.

So for routing through the layers, the routing table within the DIFs is needed to calculate the weight vector between two DIFs. If we look at figure 5-1 this means that node 3 of DIF III advertises to its lower layered DIFs that it can reach DIF I with the weight vector being the shortest path between node 3 and 1.

Before we can give a pseudo code for the routing between DIFs in algorithm 2, we first define the following extra parameters.

Definition 11
Let SP be the shortest path between a lower layer DIF and an other DIF, using the N layer DIF.

Definition 12
An adjacency matrix A, is a matrix consisting of elements a_{ij}. These elements represent the link between node i and node j. The element a_{ij} is a (m x 1) vector containing the shortest paths for m different kind of metrics.

Definition 13
Let $N_{DIFDOWN}$ be the set of all nodes which take part in the same N layer DIF and can reach the same lower layer DIF

Definition 14
Let Q be the set of nodes of the set N_{DIFUP} en $N_{DIFDOWN}$ together and let Q_{max}be the total number of nodes in set Q

Definition 15
Let $weight_{OTHERDIF}$be the weight vector in a node which gives the weight to a certain DIF

Now that we have defined the parameters, we will explain the pseudo code which is given in algorithm 2.

Algorithm 2 Pseudo code strategy 2

```
1.  SP = INF
2.  weight_OTHERDIF = ø
3.  for all metrics m ϵ M
4.      for all nodes k ϵ N_DIF
5.          for all nodes l ϵ Q
6.              weight_DIF (l,k,m)= weight_OTHERDIFIF (l,k,m) + w(l,k,m)
7.              if weight_DIF (l,k,m)< SP(l,k,m)
8.                  SP(l,k,m) = weight_DIF(l,k,m)
9.          end
10.         weight_OTHERDIF (l,k,m)= SP(l,k,m)
11.     end
12. end
13. advertise weight_OTHERDIF(l,k,m) for current node k to all other DIFs in which node k participates in
```

In line 5 to 9 all paths from a node in the DIF towards the (multiple) nodes which are connected to the same up layered or lower layer DIF are compared to each other and the shortest path between the two DIFs is calculated. The results of all the calculations is a topology matrix like in figure 5-6 with the shortest path vectors as elements. If all the shortest paths for the different metrics for that node to the other layered DIF are computed, the node advertises the weight of the path to all the other DIFs it participates in (line 13). In the text below the different lines are described individually.

Line 1: The shortest path vector is set to infinity

Line 2: The weight vector of the up layered or lower layer DIF is set on zero.

Line 3: For every metric

Line 4: For every node in the DIF

Line 5: For every node which takes part in both the N layer DIF as in another layered DIF

Line 6: The weight vector of the DIF is the weight vector of the other layered DIF in node p, plus the shortest path between node p and node k of the N layer DIF. The shortest path within the DIF is calculated by the protocol used within the DIF.

Line 7: If the weight vector of the DIF is less than the current shortest path

Line 8: Then the weight vector becomes the new shortest path

Line 9: End of first loop. For the k^{th} node, the shortest path towards the other layered DIF is calculated, by comparing all paths between node k and the different nodes l, which are both in the N layer DIF and in the other layered DIF

Line 10: For node k the weight vector to reach the up layered or lower layered DIFs is the current shortest path

Line 11: End of third loop. For every metric all paths from the nodes of the N layer DIF towards the nodes which are also take part in the same other layered DIF it is participating in are calculated and the shortest paths each node has is advertised in the other DIFs the node is participating in

Line 12: End of second loop. All paths from the nodes of the N layer DIF towards the nodes which are also take part in the same other layered DIF are calculated

Line 13: The shortest paths towards other DIFs are advertised in the other DIFs the node is participating in

To proof that with this strategy the shortest path towards the right DIF is found, we have to proof that the all the paths from one DIF to another up layered DIF are calculated correctly and compared to each other in order to find the shortest path.

Proof

Within the DIF the shortest paths between two nodes are calculated according to the routing algorithm that is used in the DIF. The current routing algorithms that can be used are OLSR, EIGRP, Babel and MAMCRA. The proof for finding the shortest paths for these algorithm is found in [RFC 3626 2003, Siva Ram Murthy and Manoj 2004, RFC 6126 2011, Kuipers and Van Mieghem 2002]. In line 6 we add up the shortest path vector between a node and a node that is also connected to the other layered DIF to the advertised weight vector of the node that is also connected to the other layered DIF. Because there can be multiple nodes in a DIF that are also connected to the other layered DIF the calculated vector has to be compared to the currently stored shortest path for that node (line 7). If the just calculated shortest path is less than the currently stored shortest path, the just calculated shortest path becomes the new stored shortest path for that node (line 8). When all possible paths between a node and the other layered DIF nodes are compared the shortest path is stored and advertised in the other DIFs the node is participating in¤

6-2-2-2 Complexity of strategy 2

If we look at the pseudo code in figure 2 we see three loops, one for the different metric, one for the nodes participating in both N layer and another layered DIF and one for the nodes of the DIF. In the worst case scenario all nodes of the N layer DIF are participating in the other layered DIF. This means that in the worst case scenario the first two loops has to run $N_{DIF,max}$ times. This process has to be repeated for every metric, which makes the overall complexity of the order $O(m_{max}N^2_{DIF,max})$. This is the same complexity as already known and used protocols in the current internet. But because the current internet is a flat network and not recursive like a RINA network, the number of nodes in a DIF is always less than in a flat network. The total number of nodes is the same, but in a RINA network these nodes are spread over different DIFs in which processes can run in parallel of each other. This means that with the same total number of nodes a RINA network is always less complex than a flat network, like the current internet.

Chapter 7

Conclusions, recommendations and future work

7-1 Conclusions

In a RINA network all layers are independent of each other. This is realized by giving the nodes within a DIF a location dependent addresses, and for the communication with the other layers independent names. For the N+1 layer this is called the application name and for the (N-1) layer the point of attachment. This means that what for one layer is called the application name, is called the point of attachment name in the layer above. Because of this multiple naming for one node, route discovery becomes more difficult. Network discovery in a recursive layer architecture as RINA is only possible if the nodes in a DIF have some knowledge of the other DIFs. We have seen that the N layer DIFs trust their N-1 layered DIFs, else they never allowed a node from a lower layered DIF to join their DIF. So the knowledge a N layer DIF has, can be put down to their lower layered DIFs as long as nothing is told of how routing is done in a DIF, so that the path can not be predicted.

We have seen that before routing information between one node and another can be done, first the right DIF/application has to be found in which the destination node is situated. We have looked at two different strategies for finding the right DIF if no route towards the destination node was already available. In strategy 1 the biggest N+1 layer DIF is joined repeatedly till a common DIF is found and in strategy 2 the up layered and lower layer DIFs advertise which DIFs can be reached via their DIF using the N layer DIF. The conclusion is that the first strategy does not scale and is therefor not suitable while the second strategy, where the up layered or lower layer DIFs advertise that a certain DIFs can be reached by using their DIF to the N layer DIFs, is more efficient, scalable and less complex than strategy 1. Strategy 2 is suitable for a recursive network.

After finding the right DIF, the shortest route towards that DIF is computed. We made a proposal for the way the nodes calculate these shortest routes through the different layers,

taking into account different metrics. This proposal is written down in a pseudo code. In this pseudo code, every possible path from each node in the DIF towards the nodes in the DIF that connect the DIF to an up layered or lower layer DIF is calculated and compared with earlier found paths to find the shortest path from a node in the DIF towards another layer DIF. This is done for every single metric. A vector of shortest paths for each metric from a node in a N layer DIF to a another layer DIF is advertised in all the other DIFs in which the node of the N layer DIF is participating.

When the shortest path towards right DIF is found, the shortest path to the destination node in the right DIF has to be found. This can be done by different current existing routing algorithms. It depends on the requirements of the users of the network if a single metric or multiple metrics are used to calculate the shortest path. After comparing different current existing routing protocols, the conclusion is that only OLSR, EIGRP, Babel and MAMCRA meet the current RINA requirements, which means that they can use multiple metrics for shortest path calculation, are able to do multicast and are scalable. Although these protocols can be used in RINA, there are still some issues. EIGRP, for example is a Cisco developed protocol and is only used in Cisco routers. Other companies which develop routers can not use EIGRP.

When doing only routing using the nodes of the current DIF it might be possible that the shortest path between the two nodes in that DIF is not the overall shortest path between those two nodes. It might be possible that there is a shorter path between those nodes using a N-1 DIF which is not the physical connection between the two nodes. But in that case, the nodes of the N layer DIF needs to know how these nodes are called in other N-1 DIFs, because when a router takes part in more than one DIF it can compare the different routes between two nodes in different DIFs and route the information via the shortest path, independent of which DIF is used. In order to know how the nodes are called in N-1 layered DIFs the node which joins a DIF has to advertises its names it is using in other N-1 layered DIFs to the nodes in the DIF it just joined. Existing nodes of that DIF than can look if they have a DIF in common and if they already have a shorter route towards that node in another N-1 layer DIF. If so the router will route via that route and advertises within the DIF that it has a shorter route. The other nodes in the DIF will see the N-1 layer route between those two nodes as a new link in the N layer DIF. Only the two nodes involved, know that this is not the case and that the routing is done by using the N-1 layer DIF.

The problem is that current routers can not do such operations. So it is better to develop a complete new open standard routing protocol. And it would even be better if the users of the network can put in their own requirements for that network. In that case this protocol would be the same for every RINA network, but could be specified for every independent RINA network or even specified for every single DIF. In this way different QoS requirements could be realized.

Finally when looking at the overall complexity of the network discovery process we found out that when using strategy 2, a DIF in a RINA network has the same complexity $O(m_{max}N^2_{DIF,max})$ as a network without recursive layers, like the current internet. But because the amount of nodes (N) in a recursive layer is always less than in a network without recursive layers, a DIF in a RINA network is always less complex than the network without the recursive layer architecture.

7-2 Recommendations

After investigating the two different strategies to find the right DIF, we would recommend to use strategy 2 for finding the right DIF. Strategy two is the most efficient as the network has a squared architecture. This means that the number of layers is as big as the number of elements of a layer. In that way the calculation time through the layers takes as long as the calculation time within the layers. In practice this is hard to achieve, as hosts will join and leave DIFs all the time.

In this thesis we formulated a proposal for finding the shortest path toward the right DIF. The next step is to translate the pseudo code into software code. If the software code is developed, simulations should be run to validate the code. Only when the software code is validated, the code can be implemented.

For the implementation of RINA, we would recommend that the introduction of RINA networks should start small. A RINA network could perfectly used as a network internally to a company and could be attached to the current internet. If more and more companies adopt the RINA architecture and start implementing RINA in their companies, the RINA architecture could expand as wildfire.

7-3 Future work

RINA is still in the development phase so a lot of work still has to be done. One of the things we mentioned in this thesis is the routing protocol. In theory a different routing protocol could be used in every DIF, but then the routing using different DIFs would became very difficult. Also smart routers has to be developed which could distinguish between routes in different DIFs and map the different names of the nodes in different DIFs to one particular node, so that the different routes in the DIFs could be compared with each other.

In this thesis a theoretical proposal of how the shortest path towards a certain DIF can be found is made. The next step is to transfer this pseudo code towards software code and validate this code by running a simulation.

In January 2013, a project group consisting of the i2CAT foundation [i2cat], Nextworks [Nextworks], iMinds [iMinds] and the Interoute [Interoute] was established for the IRATI project [Irati]. The main goal of this project group is to bring RINA to an architecture reference model. This will be done by the design and implementation of a RINA prototype on top of ethernet. This project will take two years and is still in progress.

Followers of RINA are gathered in the Pouzin society [Pouzin]. Specific information, presentations, papers and video's are chaired in the RINA web page of this society [RINA]

Appendix A

Delta-t versus TCP and UDP

In this appendix three traffic transfer protocols are compared, Delta-t which is used in RINA and TCP and UDP which are used in the current internet.

A-1 Delta-t

Delta-t is a transport protocol developed at the Lawrence Livermore Laboratory in the beginning of the eighties. The initial requirements were [Watson 1989]:

1. Minimum packet exchange for request/response transactions
2. High throughput bulk data transport and other stream services
3. Flow control without polling for reliable zero window opening
4. Error control of lost, damaged, duplicated, and out-of-sequence packets
5. Large and flexible name space for transport end points
6. Message boundary preservation
7. Secure communications

In connection management there are three logical phases: initialization, data transfer and termination. During the initialization phase error control identifiers called sequence numbers are established. In order to do that in a reliable way, no packets from a previously closed connection should be left in the network anymore. Those old packets could else give duplicates. This means that when a connection is terminated the receiver must not close until it is sure that it has received all possible retransmissions and delayed data packets. The sender must not close until it has received all acknowledgments the transmitted data. A safe interval for the sender should be at least 3(MPL+R+A) seconds, where MPL is the maximum packet

lifetime, R is the retransmission interval and A is the time it will take at the receiver before an acknowledgment is sent. At the receiver the interval of the timer should be at least 2(MPL+R+A) [Watson 1981]. If the timers are set right, it is sure that all sent packets are acknowledged or discarded and that no original packets do exits in the network anymore. A packet is discarded if the MPL becomes 0 before it arrives at the receiver.

For acknowledgments, delta-t uses a data-run flag which acknowledges all previously send sequence numbers. For example data-run flag 3 means that packet with sequence number 1, 2 and 3 are all acknowledged. An big advantage of this is that the packet exchange for request and response transactions is minimized in the same time. Next to that it labels its data bits with a protection level and optional with synchronization marks.

A-2 TCP and UDP

TCP is a transport protocol that is used in layer 4 of the internet architecture [RFC 793 1981]. Next to TCP another major protocol that is used in layer 4 is UDP. UDP is an unreliable, connection less packet service that complements the IP-protocol of layer 3 only with error-checking an application distinctions [RFC 768 1980]. In layer 3 only the header is checked and UDP checks the whole packet. Because UDP uses application distinctions, UDP is multicast capable and is therefore used in real-time multimedia applications and querying applications like DNS and ping.

TCP is a reliable, end-to-end, connection oriented transport protocol that uses the connectionless, unreliable IP network. It is point-to-point and doesn't support multicast or broadcast in principle. In TCP each connection has its own queue, so parallel data streams can take place. TCP uses flow control and automatic repeat request (ARQ) mechanisms. Flow control is done by splitting the data steam in segments. These segments are numbered so that at the destination the data stream can be reconstructed in the right order. To acknowledge a packet the receiver sends an ACK packet with the next sequence number it expects, so packet with sequence number 1 will be acknowledges by an ACK packet with sequence number 2. When TCP sends a segment it starts a timer, which is reset when the ACK is received. If the ACK is not received within the retransmission-time-out (RTO) the segment is retransmitted. At the receiver segments can arrive out of order or can be received multiple times, because IP does connectionless forwarding. Duplicate segments are discarded at the receiver and out of order segments are put into the right place. The amount of data that will be sent is controlled by a sliding window technique. Each node advertizes its maximum segment size (MSS). This represents the buffer space of the receiver, the amount of data the receiver is willing to accept at that moment. The sender can adjust its data stream accordingly.

To setup a TCP-connection the three-way-handshake is used. First the sender A requests the destination B to open a connection and specifies a port number, this packet has sequence number x. B replies with an ACK with a sequence number x+1 and gives its own packet a sequence number y. A now confirms the ACK from B with sequence number y+1 and set his own packet sequence number on x+1. When this packet arrives at B the connection has been established. To end a connection the opposite way is followed. If A wants to end the connection it sends a FIN flag to B. B acknowledge this packet and informs its application

layer process that no data from A will be received. B continues to send information it has for A until it has no longer any data for A. A keeps acknowledging this information till B sends a FIN flag, indicating that it has no data anymore for A. When A has acknowledged this flag the connection is terminated. An other way to terminate a connection is via the reset (RST) flag. The RST flag is sent if for example an RTO has taken place or when the packet can not be delivered to the next node and the crack bank procedure is in place.

The receiver of data generates an acknowledgment at the moment it receives a packet. The sender now can calculate the round trip time (RTT) for a packet. A new packet is only sent into the network until an old one leaves the network. So if enough packets are sent, the network adjusts itself to the amount of data it can handle. But at the start of a connection not enough packets and acknowledgment have been send to get a stable environment. To overcome the start, the slow start algorithm has been developed [Jacobson and Karels 1988]. Next to the advertised window where the maximum segment size is in, a second window called the congestion window W is send to the sender. At start the W =1 and after each acknowledgment W is incremented by 1 till the slow start threshold is crossed. So every RTT W is doubled, because when W=2, two segments are send and two acknowledgments are received and W=4, etc. After crossing the start threshold the congestion avoidance phase is started. During this phase W is incremented by at most 1 (1/W) every RTT and follows a linear pattern now.

In case of a lost packet retransmission takes place. Lost packets can be detected in two ways, when the RTO is exceeded and when multiple duplicate ACKs (usually 3 or 4) have been received. When a RTO occurs the sender retransmits all the packets which not have been acknowledged yet. The slow start threshold is now set to half the current congestion window and the new congestion window W is set to 1 before the slow start algorithm will start. When duplicate acknowledgments are received the sender also knows that a packet has been lost. The sender knows for sure that the packet which is acknowledged multiple times is the last packet that actually has arrived. The fast retransmission procedure will now step into place. At the fast retransmission procedure that next packet after the packet which has been multiple times acknowledged will be retransmitted right-away without waiting a time-out. After that the fast retransmission is followed by a fast recovery instead of a slow start. A fast recovery procedure goes as follows. After the receipt of multiple acknowledgments (for this example put on 3) the slow start threshold is halved and the congestion window is put back to the slow start threshold + 3. Each time another duplicate ACK arrives, W in increased by 1. At the receipt of the missing ACK, which confirms that all segments up to the retransmitted segment, the congestion window W is set to the slow start threshold and the congestion avoidance phase is started. The fast recovery phase is now ended.

Bibliography

[Ade and Tijare 2010] S.A. Ade and P.A. Tijare, Performance Comparison of AODV, DSDV, OLSR and DSR Routing Protocols in Mobile Ad Hoc Networks, International Journal of Information Technology and Knowledge Management, 2010

[Alkahtani et al. 2006] A.M.S. Alkahtani, M.E. Woodward and K. Al-Begain, Prioritised best effort routing with four quality of service metrics applying the concept of the analytic hierarchy process, Computing Department University of Bradford, United Kingdom, 2006

[Androutsellis-Theotokis and Spinellis 2004] S. Androutsellis-Theotokis and D Spinellis, A Survey of Peer-to-Peer Content Distribution Technologies, Athens University of Economics and Business, ACM Computing Surveys, 2004

[Asokan et al. 2008] R. Asokan, A.M. Natarajan and C. Venkatesh, Ant Based Dynamic Source Routing Protocol to Support Multiple Quality of Service (QoS) Metrics in Mobile Ad Hoc Networks, Electronics and Communication Engineering Department, Kongu Engineering College, Perundurai and Bannari Amman Institute of Technology Sathiyamangalam, India, 2008

[Cormen et al. 1991] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, Massachusetts Institute of Technology, United States, 1991

[Day 2008] J. Day, Patterns in Network Architecture: A Return to Fundamentals, Prentice Hall, 2008

[Fu et al. 2008] B. Fu, F.A. Kuipers and P. Van Mieghem, To Update Network State or Not?, IEEE Telecommunication Networking Workshop on QoS in Multiservice IP Networks, 2008

[Grasa 2012] E. Grasa, RINA: Update on Research and Prototyping Activities, Global Future Internet Summit, 2012

[i2cat] http://www.i2cat.cat/

[iMinds] http://www.ibbtstrategy.be/

[Interoute] http://www.interoute.com/

[Irati] http://irati.eu/

[ISO/IEC 7498-1 1994] Information technology - Open Systems Interconnection - Basic Reference Model: The basic model, 1994

[Jacobson and Karels 1988] V. Jacobson and M.J. Karels, Congestion Avoidance Control, Lawrence Berkeley Laboratory and University of California at Berkeley, United States, 1988

[Kuipers and Van Mieghem 2002] F.A. Kuipers and P. Van Mieghem, MAMCRA: a constrained-based multicast routing algorithm, Information Technology and Systems, Delft University of Technology, Computer Communications, 2002

[Kuipers et al. 2002] F.A. Kuipers, T. Korkmaz, M. Krunz and P. Van Mieghem, An overview of constraint-based path selection algorithms for Qos routing, IEEE Communication Magazine, vol. 40 no. 12, 2002

[Kuipers 2006] F.A. Kuipers, QoS protocol and algorithm join forces, IEEE Communications and Networking Conference, China, 2006

[Laven and Hjartquist 2008] A. Laven and P Hjärtquist, Multimetric OLSR and ETT, Department of Computer Science Karlstad University, Sweden, 2008

[Nextworks] http://www.nextworks.it/

[RFC 768 1980] User Datagram Protocol, Internet Engineering Task Force, 1980

[RFC 793 1981] Transmission Control Protocol, Internet Engineering Task Force, 1981

[RFC 1058 1988] RIP, Internet Engineering Task Force, 1988

[RFC 1076 1988] HEMS Monitoring and Control language, Internet Engineering Task Force, 1988

[RFC 1142 1990] OSI IS-IS Intra-domain Routing Protocol, Internet Engineering Task Force, 1990

[RFC 1157 1990] A Simple Network Management Protocol, Internet Engineering Task Force, 1990

[RFC 1189 1980] The Common Management Information Services and protocols for the internet (CMOT and CMIP), Internet Engineering Task Force, 1980

[RFC 2328 1998] OSPF Version 2, Internet Engineering Task Force, 1998

[RFC 2453 1998] RIP Version 2, Internet Engineering Task Force, 1998

[RFC 3561 2003] Ad Hoc On-Demand Distance Vector (AODV) Routing, 2003

[RFC 3626 2003] Optimized Link State Routing Protocol, Internet Engineering Task Force, 2003

[RFC 4272 2006] A Border Gateway Protocol (BGP-4), Internet Engineering Task Force, 2006

[RFC 4728 2007] The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4, 2007

[RFC 5340 2008] OSPF for IPv6, Internet Engineering Task Force, 2008

[RFC 5798 2010] Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6, Internet Engineering Task Force, 2010

[RFC 6126 2011] The Babel Routing Protocol, Internet Engineering Task Force, 2011

[RFC 6335 2011] Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry, 2011

[RINA] http://rina.tssg.org/

[Perkins and Bhagwat 1994] C. E. Perkins and P. Bhagwat, Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers, IBM TJ Watson Research Center Hawthorne New York and Computer Science Department University of Maryland, 1994

[Pouzin] http://www.pouzinsociety.org/

[Saltzer 1978] J.H. Saltzer Naming and Binding of Objects, in Rudolph Bayer, et al. Operating Systems– An Advanced Course, pages 99-208, Springer-Verlag, 1978

[Siva Ram Murthy and Manoj 2004] C. Siva Ram Murthy and B.S. Manoj, Ad Hoc Wireless Networks, Architectures and Protocols, Prentice Hall, 2004

[Van Mieghem and Kuipers 2004] P. Van Mieghem and F.A. Kuipers, Concepts of Exact QoS Routing Algorithms, IEEE/ACM Transacions on Networking, Vol 12, No. 5, 2004

[Van Mieghem 2006] P. Van Mieghem, Data Communications Networking, Techne Press, Amsterdam, 2006

[Watson 1981] R.W. Watson, Delta-t Protocol Specification, Lawrence Livermore National Laboratory, California. United States, 1981

[Watson 1989] R.W. Watson, The Delta-t transport protocol: features and experience, Lawrence Livermore National Laboratory, California, United States, 1989

[Internetsociety] http://www.internetsociety.org/internet/what-internet/history-internet/brief-history-internet

Glossary

List of Symbols

Abbreviations

ABR	Associativity based routing
AODV	Adhoc on demand distance vector
AP	Application process
APM	Application protocol machine
ARP	Address resolution protocol
ARPA	Advanced research projects agency
ARQ	Automatic Repeat Request
AS	Address Space
AS	Autonomous system
BGP	Border gateway protocol
CDAP	Common distributed application protocol
CGSR	Cluster-head gateway switch routing protocol
CMIP	Common management information protocol
CRC	Cyclic redundancy check
DARPA	Defense advance research projects agency
DIF	Distributed interprocess communication facility
DoD	Department of defense
DSDV	Destination sequenced distance vector protocol
DSR	Dynamic source routing
DVP	Distance vector protocol
EFCP	Error and flow control protocol
EIGRP	Enhanced interior gateway routing protocol
FORP	Flow-oriented routing protocol
FSR	Fisheye state protocol
FTP	File transfer protocol

HEMS	High-level entity management system
IAP	Interprocess communication access protocol
IEEE	Institute of electrical and electronics engineers
IETF	Internet engineering task force
IGRP	Interior gateway routing protocol
IMP	Interface massage processor
IPC	Interprocess communication
IS-IS	Intermediate system to intermediate system
LAR	Location aided routing
LORA	least overhead routing approach
LSR	Link state routing
MAMCRA	Multicast adaptive multiple constraints routing algorithm
MSS	Maximum Segment Size
NMS	Network management system
NS	Name Space
OLSR	Optimized link state routing
OSI	Open system interconnection
OSPF	Open shortest path first
P2P	Peer-to-Peer
PDUs	Protocol data unit
PGL	Peer group leader
PNNI	Private network-network interface
QoS	Quality of service
RaMP	Relaying and multiplexing protocol
REIP	Resource information exchange protocol
RERR	Route error
RIB	Resource information base
RINA	Recursive internet architecture
RIP	Routing information protocol
RJE	Remote job entry protocol
RREP	Route reply
RREQ	Route request
RST	Reset
RTO	Retransmission-Time-Out
RTT	Round Trip Time
SAMCRA	Self-adapting multiple constraints routing algorithm
SNMP	Simple network managament protocol
STAR	Source tree adaptive routing
TCP	Transmission control protocol
TCP/IP	Transmission control protocol / Internet protocol

TORA	Temporally ordered routing algorithm
TTL	Time to live
UCLA	University of California Los Angeles
UDP	User datagram protocol
VRRP	Virtual router redundancy protocol
WRP	Wireless routing protocol
ZRP	Zone routing protocol