Routing and load balancing in Internet of Things (IoT)

Ole Marius Steinkjer

Master of Science in Communication Technology
Submission date: July 2009
Supervisor: Poul Einar Heegaard, ITEM

Norwegian University of Science and Technology
Department of Telematics

Problem Description

The Internet of Things (IoT) will connect heterogeneous objects through many different network providers and different network technologies, providing a large variation of new services.
This heterogeneity imposes great challenges on the management of network resources and Service Level Agreements.

Telenor is developing a Connected Objects Operating System (COOS) which aims at integrating and managing the interconnection of objects providing diversified services.

The assignment includes the following tasks

1. Study the Telenor Connected Object (CO) infrastructure
2. Study existing load balancing techniques that are relevant for COOS
3. Make a model of COOS in order to investigate load balancing techniques
4. Propose usage scenarios for COOS which need performance differentiation and load balancing
5. Conduct load balancing experiments on scenarios with different techniques
6. Propose principles for load balancing in COOS

Assignment given: 15. January 2009
Supervisor: Poul Einar Heegaard, ITEM

Routing and Load Balancing in "The Internet of Things"

Master Thesis

by

Ole Marius Steinkjer

July 16, 2009

Supervisor:
Prof. Poul Heegaard
Haldor Samset

The Norwegian University of Science and Technology

Faculty of Information Technology, Mathematics and Electrical Engineering

Department of Telematics

SUMMARY

The Internet of Things (IoT) is growing rapidly. It is also expected to keep up this growth in the coming years. This rapid growth, not only in size, but also in the large variation of services, requirements and variation of equipment calls a platform to wield the future IoT. Telenor is working on a solution called Connected Objects platform. This is a layer of its own in the network stack using underlying connectivity and transport, but also adding several features.

This new layer calls for its own routers, routing policy and specialization to handle the added features. The Connected Object Operating System (COOS) routers are developed for this task. These are purely software based with both the pros and cons this involve. One of the cons is the performance. The delay through these nodes greatly increase as the traffic increases. This performance issue on the COOS routers decreases the effect of load balancing across several underlying links. Adding router nodes to split the load across several routers do not give a convincing result. The added hops each packet does due to the restriction to a gateway being connected to only one router decreases the performance drastically. The most effective manner, both on reducing the delay, but also robustness on handling failures, is shown to increase the mesh ratio of the network.

FOREWORD

This report is my final work through my education at NTNU. The mix between network architecture and the performance that follows illustrates my main interest in my field of study. This is something that shines through both on courses, several internships through my studies and my choice in finishing projects and thesis. Telenors development early caught my interest. Both the expansion in internationally, but also the technological services they have developed during the years.

The development and the thought that “Everything is on the net,” is both something I have been looking forward to and also something that has astonished me. Will my food, my pets and if we go a few years back, even myself always be connected? With the platform Telenor is working on along with the growth in connectivity around the world makes it all more plausible. Not in a distant future, but today!

I would like to thank my professor Poul Heegaard at NTNU for the technical support throughout my work with the project. Both his guidelines and reassurance during my work has been motivation to continue.

I would also like to thank Haldor Samset at Telenor and Knut Eilif Husa at tellU AS. They has been very helpful in both answering vagueness’ and supporting me with access to critical data needed for my research.

Ole Marius Steinkjer

Contents

List of Figures

List of Tables

ACRONYMS

- AS is short for Autonomous System. An AS is a collection of connected IP routing prefixes under the control of one or more network operators that presents a common, clearly defined routing policy to the Internet. [20]
- CO Connected Object
- COOS CO Operating System. A deployed system running on the platform Telenor is developing.
- DICO Deployed Infrastructure for Connected Objects. A system containing of several COOS on Telenors platform.
- FQ is a load balancing algorithm called *Fair Queuing*.
- GPS Generalized Processor Sharing is a term within computer science of how several entities share the same resource.
- HI Host Identity. An identity tag in COOS.
- HIT Host Identity Tag. An identity tag in COOS.
- IoT Internet of Things
- Posix Portable Operating System Interface for Linux "POSIX (pronounced /'p?z?ks/) or "Portable Operating System Interface for Unix" is the collective name of a family of related standards specified by the IEEE to define the application programming interface (API), along with shell and utilities interfaces for software compatible with variants of the Unix operating system, although the standard can apply to any operating system. [7]
- RR Round Robin balances load on different resources packet based where each resource gets one packet each in turn.
- URI Universal Resource Identifier. An identity tag in COOS.
- WFQ Weighted Fair Queuing. A load balancing scheme like FQ, but with possibility to weight different connections.
- WRR Weighted Round Robin. A balancing policy like RR, but with possibility to weight connections.

1 INTRODUCTION

The following years we will see a drastical growth in connected objects. As [17] shows, the number of connected objects and the value-adding applications following will increase the coming years. Analysts see several changes to cope with this new trend. The connected devices will more and more be connecting through different connections, also be connected via different technologies at the same time. Users will demand more agile, adaptable and available platforms for solution development to again meet their tailored demands. On the top of this, connectivity and bandwidth will only become cheaper and cheaper. This low cost will, as early 90s futuristic visions predicted, get your refrigerator, groceries and toaster online.

Both networks and applications to handle this growth in both size and variation is needed. Telenor is launching a Connected Object (CO) platform in the near future. It is designed to work as a middle layer in the network, creating another graph structure [2]. In addition, the platform seeks to ease deployment of services through ease of addressing, standarized components and Application Programming Interfaces (APIs).

On a shared system like this there will be a large difference in the QoS demands from the different customers. Some customers might need as good as real time updates and minimal packet loss where others barely needs connectivity. Some sort of differentiated services [15] is needed to shape the traffic. To cope with the growing customer base, the network also needs to be scalable. How the network will be affected by growth in pure traffic growth, geographical and structural size or if it can scale with new technologies are all important questions in this new platform.

The focus of this paper however, is at the core of the CO platforms network. The focus in my research is on load balancing and the topology within networks like this platform (see Figure 1. I will investigate different load balancing strategies and look at the routing policy used in COOS. Comparison on the different policies up against variation of network topology is done to look for an effective solution. To investigate different effects a simulation will be run on different topologies with different settings. As little to no assumptions is made on traffic pattern, two case scenarios is set up to easier give a realistic traffic flow compared to a total randomized. The cases are set to generate traffic with a variety in arrival intensity, QoS demands and forwarding.

There are several points which could be of interest on this topic, that is not

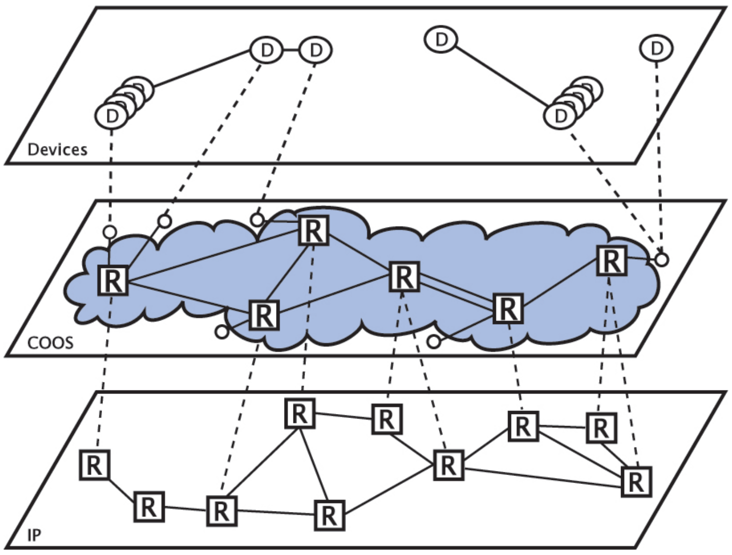

Figure 1: The main area of focus in my research.

handled in my research. Gateways connecting COOS routers to underlying network segments are not investigated. The reason for this is that litle work has been done from the projects part on these, and their behavior is still unknown or unavailable. Load Balancing from the component to different gateways is of the same reason excluded from the research. Load balancing can also be done between identical service platforms, which falls outside the scope of my research.

2 LOAD BALANCING

Everyone is affected by load balancing as good as every day. It can be calling to a call center where several callers are divided between several operators. Another example is when driving on a multilane road home from work. Here all the cars are divided on the different lanes, some may be for for all cars and some only for public transport. It can also be several friends wanting your attention in a chat application where you have to divide it in some manner.

In IKT systems, load balancing also plays an important role in making things run smoothly. A personal computer normally has only one processor (at least until the latest years). On each computer there are, however, several applications competing for the processors time. Different operating systems have different solutions to most fairly divide the resource. Each with different pros or cons making it suitable for different systems requirements. [18]

Electronic communication is also highly colored by load balancing. Cellular telephone technology takes use of both sharing of time slots, on different frequency domains and by use of different codes. In traditional network topology load balancing is also important. In a web of network links connecting different routers, each link may have different delay, capacity and other characteristics. **TODO: Hvilke tre ord bruke her?** Some links or routers might be congested. This can for example be solved by moving some load over to other resources following defined rules. These rules can tell IP packets to follow a certain path through the network or choose certain routers as a next hop.

Max-min fairness is a term in load balancing which refers to which resource is chosen next. If max-min fairness is obtained, the resource with the lowest rate is always chosen next.

Not a *load balancing* technique, but should be mentioned as a property of a scheduler. One achieves *Min-max fairness* when maximizing the lowest rate. [5]

Generalized Processor Sharing (GPS) [16], is not an a method that can be implemented. It is a method which achieves an exact weighted max-min fairness. GPS is therefor used for comparison up against other load balancing techniques to examine how effective they are. GPS is on the other hand a fluid model assuming infinitesimal packet size. Variation in packet size will not be handled by the plain GPS. Packet-by-packet GPS (PGPS) is developed for these cases [3].

2.1 Round Robin

(RR) is one of the most basic load balancing techniques one can implement and is explained in 1. It basically traverses through its resources sending one task to each resource before moving to the next. The method does not take packet size, inequality on resources or other aspects into consideration. Using RR on two links with different capacity, delay or other characteristics can because of give a suboptimal performance. Using RR can also give dynamic imbalance if packets size varies greatly.

Algorithm 1 Round Robin

```
1: while true do
2:     $i \leftarrow (i + 1) \bmod n$
3:     Return $R_i$
4: end while
```

2.2 Weighted Round Robin

Weighted Round Robin (WRR) is one of the extensions to normal Round Robin. The way WRR works is that each resource branch of the load balancer is assigned a weight. The weight will represent a resource capacity in the form of processing power, bandwidth or similar depending on the system. Load will be divided between the resources depending on the weight of each of them. Regular *Round Robin* can be seen as a branch of WRR where all weights is equal.

- $R \sim$ Resource set $R = \{R_0, R_1, \ldots R_{n-1}, R_n\}$
- $W(R_i) \sim$ Weight of resource S_i
- $i \sim$ Resource counter while running. Initiated with $i = -1$
- $cw \sim$ Current Weight
- $max(R) \sim$ Highest weight in resource set S
- $gcd(R) \sim$ Greatest common deviser of the weights in resource set S

Algorithm 2 Weighted Round Robin (WRR)

```
1: while true do
2:     $i \leftarrow (i + 1) \bmod n$
3:     if $i = 0$ then
4:         $cw \leftarrow cw - gcd(R)$
5:         if $cw \leq 0$ then
6:             $cw \leftarrow max(R)$
7:         end if
8:     end if
9:     if $W(R_i) \geq cw$ then
10:        Return $R_i$
11:    end if
12: end while
```

Algorithm 3 Greatest common diviser

```
1: if $a = 0$ then
2:     Return $b$
3: end if
4: while $b \neq 0$ do
5:     if $a > b$ then
6:         $a \leftarrow a - b$
7:     else
8:         $b \leftarrow b - a$
9:     end if
10: end while
11: Return $a$
```

Fair Queuing - Example walk through										
Packet Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Packet Weight	2	2	4	6	2	2	2	4	2	2
Link 1	2/2			6/8						2/10
Link 2		2/2			2/4	2/6		4/10		
Link 3			4/4				2/6		2/8	

Table 1: Fair Queuing dividing ten packets, step by step. Allocations are symboled by *Allocated packets size / Total allocated*

In a system with uneven resources, WRR can increase the efficiency of the system. It has on the other hand the same problem as RR, where several high-load demanding tasks go to the same resource. **TODO: known mean packet size in http://www.cs.berkeley.edu/ kfall/EE122/lec27/sld006.htm ??**

2.3 Deficit Round Robin

Load balancing is often several connections competing for one resource as explained earlier. Deficit Round Robin (DRR) is a modification to WRR which frees it from knowing the mean packet size to achieve min-max fairness [8]. In my research, the packet size is not taken into consideration so I will not examine this method more closely.

2.4 Fair Queuing

Fair Queuing (FQ) is a bit-wise load balancer compared to RR which is packet based. Based on the amount of data, or time it is occupied by a path, the information is shared between the different links.

It is easiest explained with an example illustrated in Table 1. Say we have ten packets in the buffer with sizes *2, 2, 4, 6, 2, 2, 2, 4, 2, 2.* and the counter for each outgoing link is equal (zero for ease). The next packet will always be sent to the link which has received the least amount of data. Each links local value can for example be solved with a leaky bucket, decreasing the "load recieved" at a given rate.

Short time fluctuations will of course occur, but these are a lot smaller then in RR. In addition, FQ can prevent the situation where one link gets all the small packets and one gets all the large ones.

2.5 Weighted fair queuing

Fair Queuing does not handle the weighting between the connections to load balance. As with Round Robin there is a solution for skewed distribution. Each connection is assigned a weight representing how large portion of the load it should get. The load of the i'th connection will then be:

$L_i = \frac{L \times w_i}{w_1 + w_2 + \cdots + 2_N}$, where L is the total load to be balanced.

[19] shows that WFQ can guarantee an end-to-end bound, but also used to control the QoS through dynamically changing the weights.

3 LSAROUTING

Link-State (LS) algorithms calculate the route from a source based on globally known information [12]. This is done by having each node broadcast its link-state through the network segment it belongs to. This way, all nodes will be able to calculate the best paths to all destinations in the network.

Idea behind Link State Routing is that each router must do the five following things:

1. Discover its neighbors and learn their network addresses.
2. Measure the delay or cost to each of its neighbors.
3. Construct a packet telling all it has just learn.
4. Send this packet to all other routers.
5. Compute the shortest path to every other router. (Djikstra's can be used for this as will be explained in 3.1.)

3.1 Djikstra's Algorithm

As pointed out in the list above, each node computes the shortest path from itself to each node in the network from the distributed information it has received. A widely used algorithm for this calculation is Djikstra's Algorithm [6] which solves the *single-source shortest-paths problem* on a weighted,

Algorithm 5 Pseudocode for Djikstra's Algorithm.

```
for all Nodes $n$ do
    $n.dist \leftarrow infinity$
    $n.previous \leftarrow NULL$
end for
while $N$ not empty do
    $n \leftarrow$ node with smallest dist
    $N \leftarrow N - n$
    for all Neighbor $m$ in $N$ of $n$ do
        $altRoute = n.dist + e_{n \to m}.cost$
        if $altRoute < m.dist$ then
            $m.dist \leftarrow altRoute$
            $m.previous \leftarrow n$
        end if
    end for
end while
```

Dijkstra's Algorithm - Walk through

		u		v		w		x		y		z	
Step	Removed	d	p	d	p	d	p	d	p	d	p	d	p
0		0	u	∞	-	∞	-	∞	-	∞	-	∞	-
1	u	0	u	5	u	3	u	∞	-	3	u	∞	-
2	uw			4	u	3	u	∞	-	3	u	∞	-
3	uwy			4	u	3	u	3	y			3	y
4	uwxy			4	u			3	y			3	y
5	uwxyz			4	u							3	y
6	uvwxyz			4	u								

Table 2: Dijkstra's Algorithm executed on Figure 2, step by step.

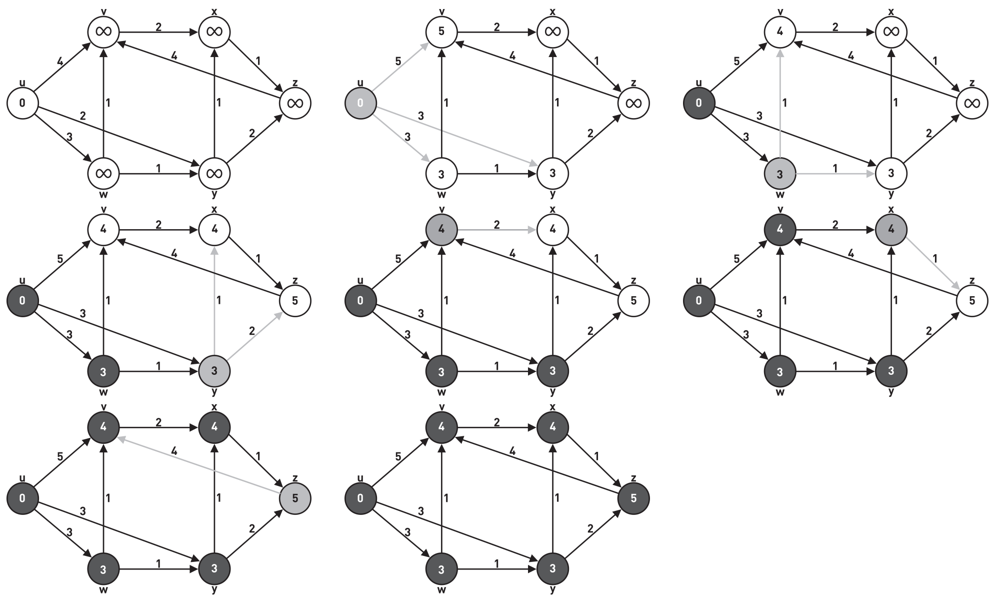

Figure 2: Step-by-step illustration of Djikstra's Algorithm during calculation.

directed graph. The algorithm repeats itself until all nodes have been calculated. After k iterations of the algorithm, the least cost paths of k nodes are known.

As we see in Figure 2 Djikstra's Algorithm is a greedy one. It always chooses the node with the lowest cost, that has not yet been calculated from. For each node in the network the algorithm calculates line 8 - 14 in Algorithm 4.

4 COOS

The trend of this decade has been that more and more people are getting connected to the Internet. The next step, which we are already seeing, is that more and more objects are also coming online. This expected explosion [17] of Connected Objects (CO) connecting to the Internet of Things (IoT), requires a system to handle this. This will be everything from addressing, QoS, giving a standard API, gathering it on one platform and so on. The expected growth in this field is far to rapid to wait for a standardization with todays lack of Standards Development Organisations (SDO) and the delay of this kind of process will take.

This growth causes opportunities for new markets and is a potential of growth

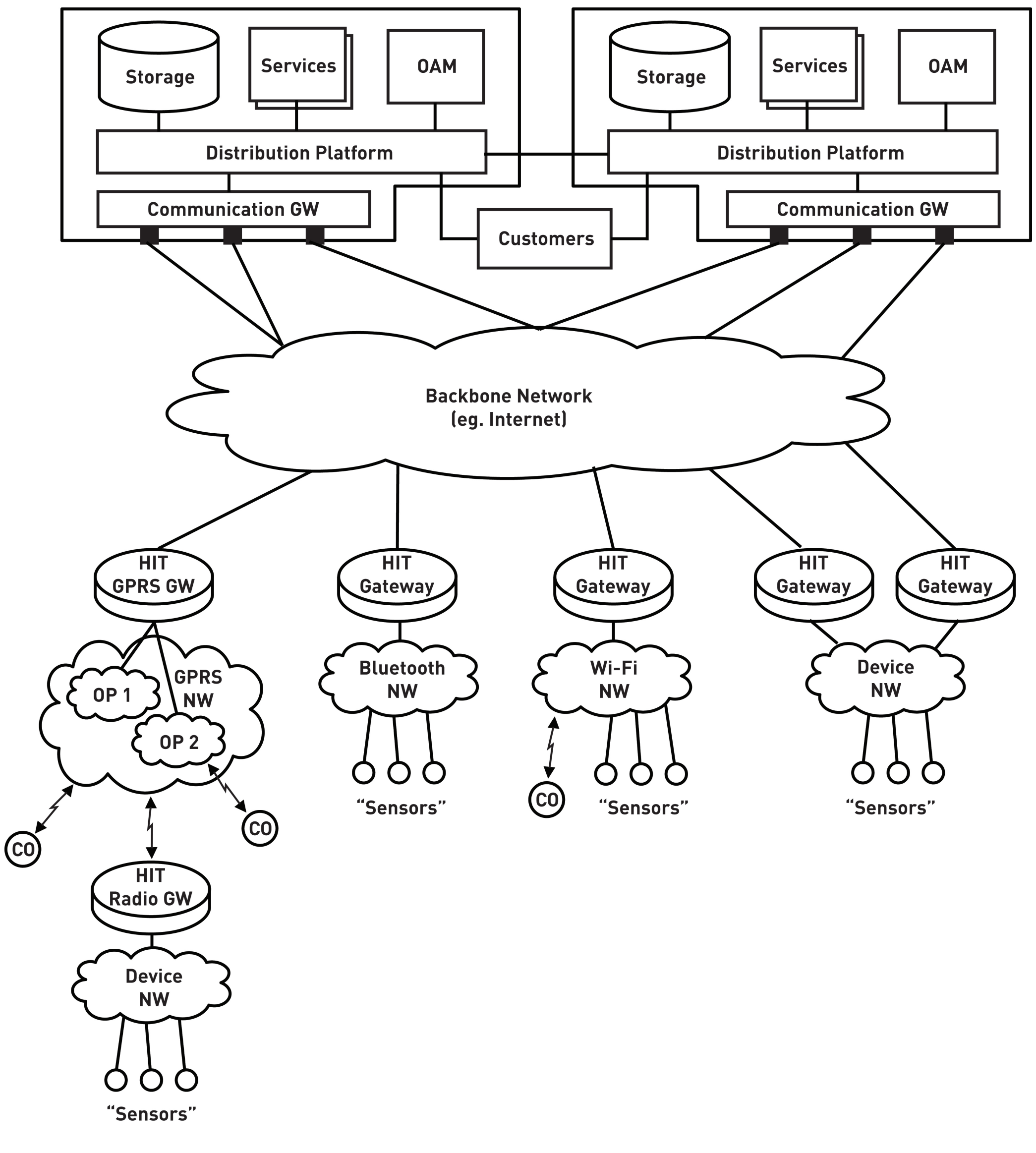

Figure 3: COOS should be able to connect objects regardless of what underlying connectivity type is used.

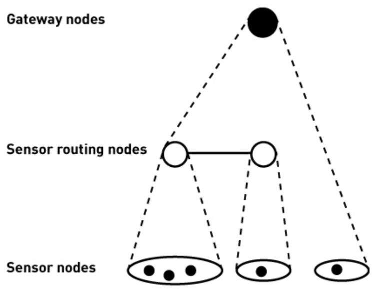

Figure 4: CO-leaf as used in the CO platform

in existing markets. To utilize this, Telenor is working on a new platform aimed at Connected Objects.

Telenors CO platform is a solution that will allow consumers to more easy deploy solutions to "The Internet of Things". It can bee seen as a new layer in the OSI model, both handling problems and giving more opportunities then the existing layers does. In addition, and probably most important, it gives a flexible, agile, scalable and technology independent architecture.

Several problems are solved in Telenors solution. I will address some key points of relevance to my research. The terms used in COOS is explained in Table 3. The documentation gives a thorough explanation of most of most subjects it handles. Unfortunately, deployment, network structure and behavior is not available at this time. Some details and statistics are left out of the report due to company policy.

4.1 Addressing

A challenge with Connected Objects (CO) is that they often will be mobile within one or several access networks and segments. The number of objects will most likely exceed the possibility of giving each object a unique IPv4 address. It is also very unlikely that a ubiquitous IPv6 connectivity is established in time to support the addressing. To cope with the lack of address space in IPv4, HIT gateways are allocated a unique IP-address. CO-leafs are then introduced to gather several objects under an addressable gateway (see Figure 4). This way, a potentially large group of COs, under the control of a single HIT gateway, can be addressed by the use of a globally unique IP-address.

The identifiers *Host Identity* (HI) and *Host Identity Tags* (HIT) are mathematically generated for objects which are the used unique tag for the objects.

COOS Definitions	
Device	This is a physical unit that is able to communicate in some way through some network.
Edge	A connection between an object and the platform, the glue that makes the device able to communicate with the whole system.
Object	This is an entity that is able to send or receive data through some network, this entity can be either a device or a service.
Service	Something that offer function(s) accessible by users through some means of communication, usually run on a device. A service can be composed of other services. The CO platform will be able to offer services.
Plugin	A Plugin is an Object that can be plugged into the Messaging bus. Object and Plugin are in many cases equivalent notions and used interchangeably.
Endpoint	The entity that the Messaging bus receives messages from and delivers messages to. Endpoint is the interface towards the Message bus of a Plugin.
Channel	Channel is the connection between Plugins and the Message Bus. It consists of Links and Transport
Link	A Link is a uni-directional path in a Channel consisting of Filters
Transport	A Transport is a means for transporting messages over an underlying medium.
Filter	A Message Processor that filters messages that flow though. Filters are parts of Links.
Processor	An entity that processes messages.
Module	A Module is a set of functionalities in COOS. A Module can be implemented by a set of Plugins and underlying Modules. Security and Messaging are examples of COOS modules
Component	Same as Plugin and Object

Table 3:

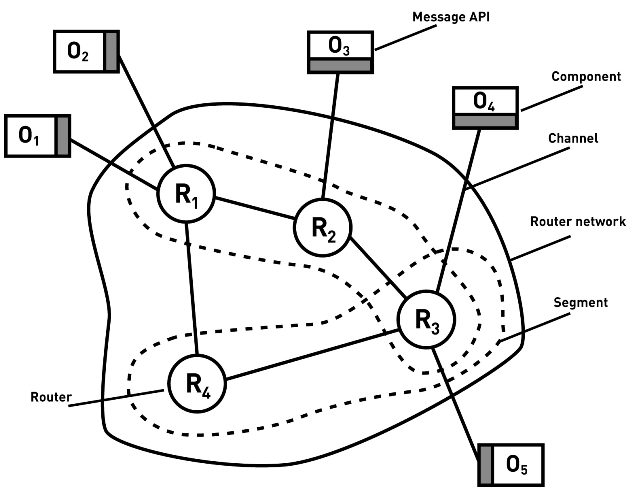

Figure 5: The CO platform contains of several routers divided in segments for addressing. Components are then connected to the routers, often via gateways.

These are generated so that no human readable information is conveyed. *Universal Resource Indentifiers* (URI) are introduced as a human readable identifier for objects. Translation to locate a object is:
$URI \rightarrow HI \rightarrow HIT \rightarrow IPaddress$
The reverse resolution may also be needed in certain cases:
$HIT \rightarrow HI \rightarrow URI$

4.2 Architecture

One of the most important features of the COOS layer is forwarding of packets through the network. Through HITs messages needs to reach the right service platform from the sensors, the other way around or between different DICOs or COOS'. The messaging service is an overlay network, a virtual network, which uses an underlying network for connectivity as illustrated in Figure 1. The service is distributed, but is centralized from the components view. The routing information is local information at each router node and is updated through LS routing. The routing information also contains information about which gateway to forward packets through, which again is connected to one of the routers. Gateways is, at this point, thought to be connected to a router in a one-to-one relation, where objects may be behind several gateways. An example can be a device connected through GPRS and regular IP at the same time.

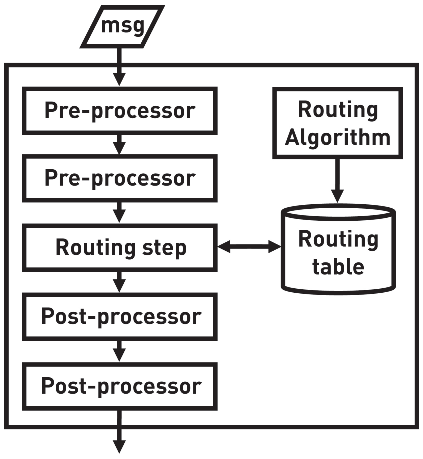

Figure 6: COOS router processes

The COOS network architecture will contain of developed COOS routers. This is basically a system running Java on any supporting platform. Each packet will be processed asynchronously at the router illustrated in Figure 6. The pre- and post processes can be, among others, *logger*, *delayer*, *TokenChecker*, *LoadBalancer* and so on. All these are configured at the router manually.

Each router contains channels which represents a unidirectional TCP/IP, or similar, connection to another router, gateway or a COOS component. These channels are made up of an inbound and an outbound link, which again are made up of different filters. The filters are set up on configuration and will affect different packets differently. One filter is for example a *Guaranteed Delivery* which will retransmit packets with the correct flag if the channel is disconnected. The router tries it a certain number of times every 30 ms. and notices the sender if it fails. The transporter is the component which handles the transport of the messages. As we can see in Figure 8 it is on outgoing traffic, through the channels, the priority flag comes to effect (zone D in the figure). At the writing moment, no solution for prioritizing messages within the forwarding step is done.

In the router itself (zone C in Figure 8, each passing message is handled as a unique thread. All of these run in parallel. Zone A represents the underlying connectionless network where the order of messages are not guaranteed and therefor messages arrive in a randomized manner. The in channels, zone B, are First Come, First Out (FIFO). A thread for the incoming packets are generated in order as they arrive the router.

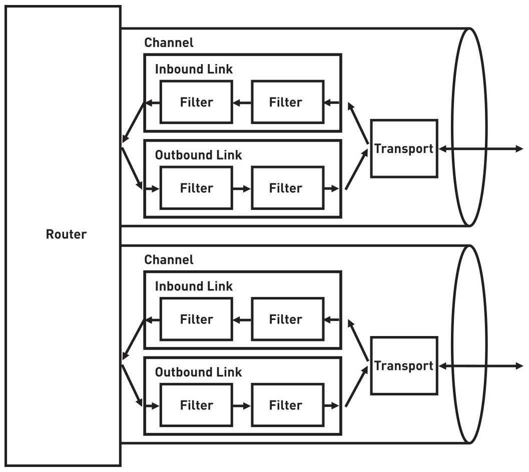

Figure 7: Each router contains of channels which again is made up of filters, links and a transporter.

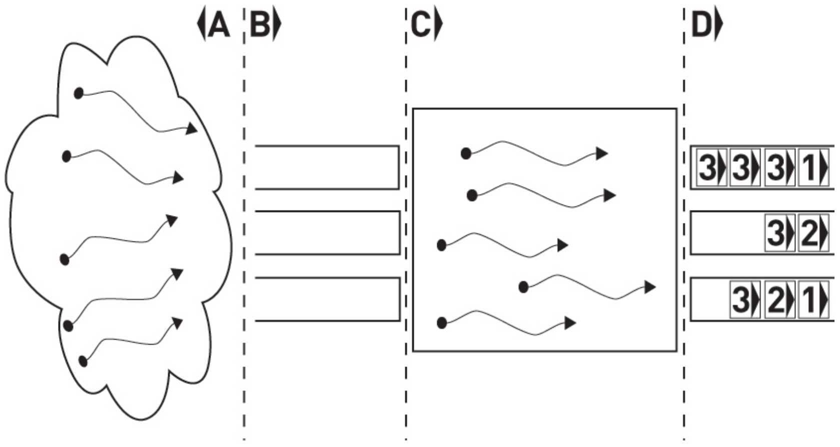

Figure 8: Messages going through a router is handled differently depending on where in the router they are.

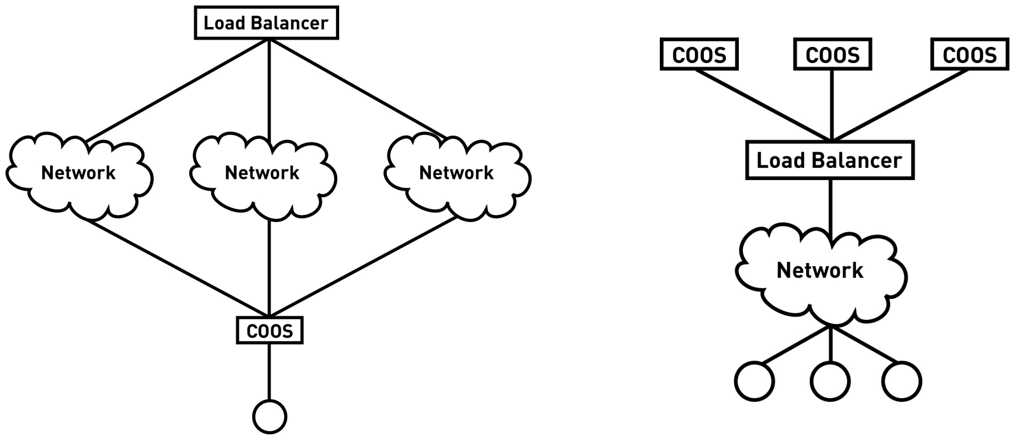

Figure 9: The Load Balancing component as thought in two different situations of use.

Load Balancer In the CO platform, there are several components defined, which again are grouped together. A COOS instance can define which of these components to inherit. The details of all of these components are outside the scope of this thesis. A *Load Balancing* component is one of these. Little information about how this component is going to work is available. It is optional and it uses Round Robin to load balance. The resources to load balance between is predefined during configuration.

Figure 9 shows the methods thought as published. The *Load Balancer* should in one scenario be able to divide load between different network technologies. This could be a PDA with access to 3G and wi-fi where the traffic should be divided between these two. Another scenario could be an identical service existing on different components or service platforms. The *Load Balancer* will then balance the load between these. Information where this *Load Balancer* will be located is either unavailable or not set at this time. From the published figures, similar to Figure 9 a set point of where it is located is not set. With distributed service platforms to load balance between, locating it in the routers could be argumented for. For balancing within a service platforms resources, a local *Load Balancer* might be the most effective. Thorough enough information about the objects and service platforms to research the effect of load balancing was not available in my research.

5 SCENARIO

TODO: Import figures

When simulating unfamiliar behavior, case scenarios is useful to set variables and behavior. This comes to everything from number of customers, QoS demands and behavior. The two cases that is used during simulation gives a variation in count, demands and retransmition rate among others. Variation in packet size is not regarded in these scenarios. Only one test setting is done on the routers with change in packet size at a locked configuration and load. This is not enough to run a plausible simulation without it being to colored by assumptions.

5.1 Case 1 - Animal herds

Setting herds out on bait in the wilderness is common in Norway. During bait however, farmers have little to no control over their herds. This can be position, if they are stuck or died for some reason. As coverage of cellular telephone technology is closing in on complete coverage, this can be used as an access medium for sensors herds carry around.

Traffic pattern As the deployed sensors should be as small as possible, calculations will be done at a server side. A timer based reporting will be most natural to use, both for power saving and ease of configuration. All communication will also be from the carried devices to central servers. A total of 26500 generating entities are used in the simulation with given traffic sinks to reach from set gateways. Each entity represents an animal reporting for the herd it is in. Each entity sends an update with given time slots of 15 minutes.

Time demands The herds will be on bait, often far from civilization. Real time demands will have no effect as the farmer will not reach the animal quickly anyway.

Packet loss Of the same reasons as above, packet loss in the COOS network will not be critical. Actions should be handled if several packets are lost in a row. I will not handle actions on continuous packet loss in my experiment, which can more easily be solved at a higher level.

5.2 Case 2 - Patient supervision

In todays systems, home alarms has to be manually triggered for a community care nurse to be notified. It also works only in the home of the patient so she or he will be forced to stay within their home. An automatic update of the patients condition and position will in a larger degree guarantee for the patients need of help. In addition it will give the patient the possibility to move outside of its home. This scenario is in addition split up in two different traffic types. One is the continuous update messages of the patients condition. If an alarm should be sounded, based on the patients information, an alarm should be sounded to the nurse.

5.2.1 Case characteristics - Patient notifications

As the devices carried by the patients are portable, they should be as light as possible excluding computation power as much as possible. The data sent will then be clear values like position, hart rate and similar values needed to see a patients condition. The notifications are tested

Traffic pattern These messages started to be similar to be alike the ones in Case 1, a set interval based repetition. Each message containing information about the patient at the given time. The interval will on the other hand be a lot shorter, as time is in a much larger degree important. As this generated to little variation, these messages was rewritten to have a variable waiting time. This time was modeled as a poisson process with the mean value the same as the static earlier value.

Time demands As every second will count when saving lives, the time demands should be as strict as possible. The time demands should however be loose enough that it can handle high load nodes to a certain degree.

Packet loss Of the same reason as above, packets should have high priority and guarantee, where possible, a delivery.

Case values			
Case	Alarm messages	Patient messages	Animal messages
Type	1	2	3
Priority	1	2	3
serverToClient	TRUE	FALSE	FALSE
Retransmit Timer	Variable	Static/Variable	900000 ms
	Avg: 1 alarm/2 min.	Avg: 15s	
Guaranteed Delivery	TRUE	TRUE	FALSE
Time demand	5000	15000	90000
QoS routing choice	Delay	Delay	Hop Count

Table 4: Difference between packet types and their QoS values used in simulation.

5.2.2 Case characteristics - Alarm messages

When a patient either hits an alarm button, his pulse raises to the roof or some other event that should generate an alarm, the community nurse should be alarmed. This alarm is sent from a Service Platform to the nurse to inform him or her about the patients status, whereabouts and so on.

Traffic pattern These will have no repeated pattern as the information messages. We look at the different alarm situations as independent of each other and model the intensity of these messages as a poisson process. They will also occur relatively seldom. I started out with an alarm each 15th minute and raised it up to each 2nd minute to look for difference.

Time demands Of the same reasons as above an alarm message should reach the destination within a strict limit.

Packet loss A loss of packet will be critical in this case so packet loss should be avoided at all means.

6 SIMULATION

At the writing moment the simulated system is only at a test phase. Tests have been mainly isolated with a goal to get specific performance values. To be fully able to to see the effect different topologies and load balancing has on the system the solution needs to be set in a context. I came to the point that only a simulation would produce these results.

TODO: add reason of simulation TODO: Add things about modeling system from MAD and Lilja. To complex to model and therefor simulate.

6.1 Goals of simulation

The COOS system is not yet put into use and is therefor not tested throughly. This counts for all parts of the system. There does however exists performance tests on delay through routers according to load on one single router with variation in the number of threads. The information about the system available is mainly restricted to the routers as explained earlier. The tests executed on the routers show us that the performance is colored by a high level language. The delay through the routers greatly exceeds the performance experienced in lower layer architectures.

Employees at Telenor have helped me with the effect several parallel computations have on the delay through each hop. The statistics are output from performance tests run by [21]. The tests still lack some detailed information for optimization. More details around when the thread count variates along with closer details on which tasks in the router takes time are recommended to fully simulate the behavior. In addition, the effect variation of packet size has on the performance is not thoroughly tested.

In the finished COOS platform, messages will have different choices on which QoS parameters to be routed by. The model I have created supports this, but is limited to route based on hop count and total delay (in ms.). This is done due to, again, lack of information about the system. With more system tests to reveal system times, behavior and delays on link or node failures and a clarification on link loss ratios this aspect could more correctly be examined. A plausible model on how the architecture of the system with respect to number of routers and links between them and how and where the gateways are located along with the performance of them should be identified.

6.1.1 Simulation tools

There are both network simulators and general simulator languages at the market. Depending on the task to simulate, different ones are more suitable then others. The ones regarded in my research and the reason of my choice is explained shortly.

Simula/DEMOS Discrete Event Modeling on Simula (DEMOS) is a package written in Simula [1]. Simula by itself is sufficient to compose any simulation model, but leaves it to the user to define all the boundaries. Simula on the other hand does not give any help with building blocks, which the user has to define.

DEMOS is based on *entities* competing for *resources*. It gives the building blocks for this in addition to some statistical help.[4]

JavaDEMOS This package for Java is based on the same packet for Simula. It's main difference is that it is a package for Java. As with Java, Software Development Kits for the language is also more evolved then its predecessor. Modeling using this package would most likely be more effective. However, experiments show that larger experiments meet problems in simulation efficiency. [9]

Network Simulator 2 (NS2) As the above, NS2 is also a discrete event simulator. It is targeted at network behavior. The simulator has considerably support for TCP, routing and multicasting protocols over different network technologies.

The simulator compiles under C++ and is developed on several kinds of Unix distributions. It should also run on any Posix system in addition to Windows.

As mentioned, NS2's main focus is on the transport and known routing protocols. The system I am investigating is in a larger degree on top of what NS2 offers, with several special properties. Modeling the system might be easier in NS2, but giving the different parts the special properties they need, would most likely need severe scripting. [11]

6.2 Simulator logics

Simula/DEMOS was seen as the most suitable simulation language to model the system in. This simulator is mainly operated as several entities competing for resources. I modeled the system with the logics laying in a packet entity

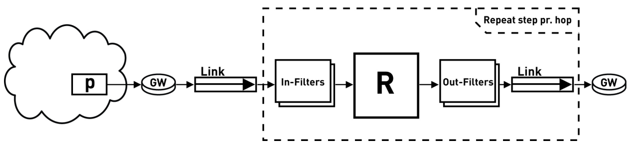

Figure 10: Simplified logics of a packet. The stippled tasks are done for each hop the packet does through the network.

alone. As can be seen in Figure 10 it can shorty be seen on as a while loop running as long as the packet has not either reached its destination or a break threshold is breached.

Figure 11 illustrates the logic of the packet entity in more detail. As can be seen, packets are only generated at the beginning of the simulation and reused by reseting the packet. This is done for performance issues. Several details in the COOS solution are not modeled as it would not affect the performance or the detail of information is not available. The throughput tests Telenor has performed on the system, gives basis for the statistics in the model. The number of threads each router runs, affect the performance each packet experiences. The performance tests of the system characterizes the delay variation in three groups. These numbers gives basis for a division into more groups to take smaller variation into account.

Entity solution In my solution I have kept the number of different entities at a minimum. This because as synchronization between entities affect performance drastically. In addition, creation and termination, is kept to a minimum. As far as it is possible, updating of static memory variables is used. This prevents the simulator to use a variable memory size during simulation, which also affects performance due to segmentation [18], among other factors.

The main entity, *packet*, has most of the logics in the simulator. For each step through the network it handles forwarding, link choices, COOS specific behavior like *Guaranteed Delivery* (see Section 4), handling of broken links and statistics. Each entity is only created at simulation startup and is kept alive through all of the simulation. Each packet entity represents a patient or animal herd in the case scenarios. Each time an update is sendt, the representative packet entity is triggered. This way the memory usage is kept fairly static and most of the memory access done is reading from and writing to existing variables.

Routers are not modeled as an entity. Synchronization cost is as explained

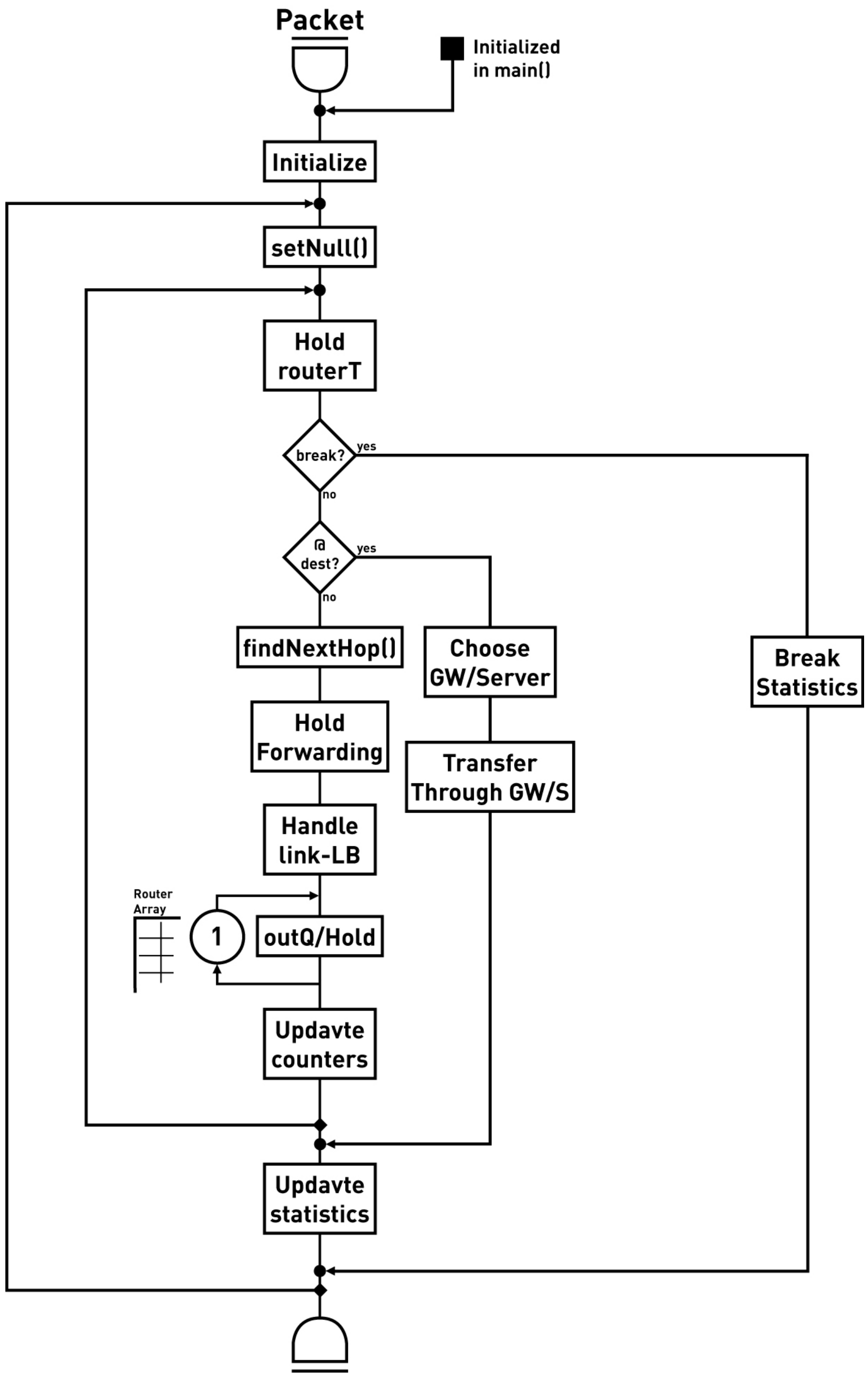

Figure 11: All of the router logics is placed into procedures in the packet entity of performance reasons. The packets are simulated according to statistical defined boundaries.

the reason for this. The local information needed about the routers was also possible to keep in static memory variables. All transport costs, routing topology and link failures are kept in globally accessible matrices. In practice this information will be stored locally. The performance of LS routing protocols is outside the scope of this research and therefor not modeled. Routing logics is handled by a procedure which is again called upon each time the topology changes.

Control over and knowledge about them *links* in underlying network layers is difficult to do dynamical. In addition, these are looked on as fairly static in performance. These are therefor modeled only as a delay between hops in the network. Delay and loss tests over a longer period between the major cities in Norway was performed during the research. Simple tools like *ping* and *tracert* was used to give a plausible topology to simulate. The connections in the CO network will also be connection oriented. This will decrease the variation even more. The delay through the underlying network is therefor set to a static value. Figure 12 shows the tested delay times experienced through the simulated network.

The *transporter* on the other hand is modeled as a resource packets will compete for. It is, as explained in section 4, in the transporter the priority of the packets is taken into account. Exact performance measures of the processes in the router is not tested. System times are calculated out from throughput tests done by Telenor, but they do not measure how effective the different processes are. This affects the accuracy of the simulation. Change in these values affect the outcome slightly.

6.2.1 Delay in router

Telenor has run several tests on the delay on packets running through a router, varying the number of parallel threads running. The tests have been run on 10, 50 and 100 threads in addition to 50 threads with 1kB size on packets. Each test is presented as an empirical distribution, where the delay results are grouped in bulks according to a time interval delayed through the router.

The statistical information of on delay through routers available is already treated and visualized. Therefor we have to make assumptions of the behavior of these times when number of clients changes. On the basis of the histograms for 10, 50 and 100 clients, statistical probabilities were be made.

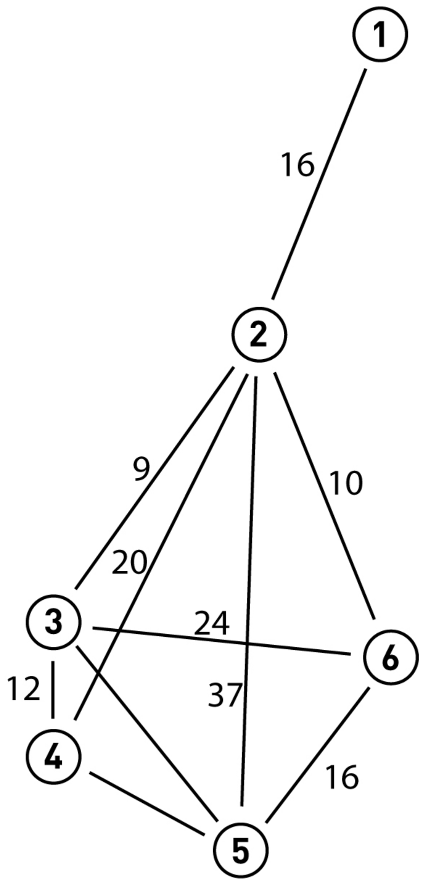

Figure 12: The delay in ms. in the topology the simulator was performed on.

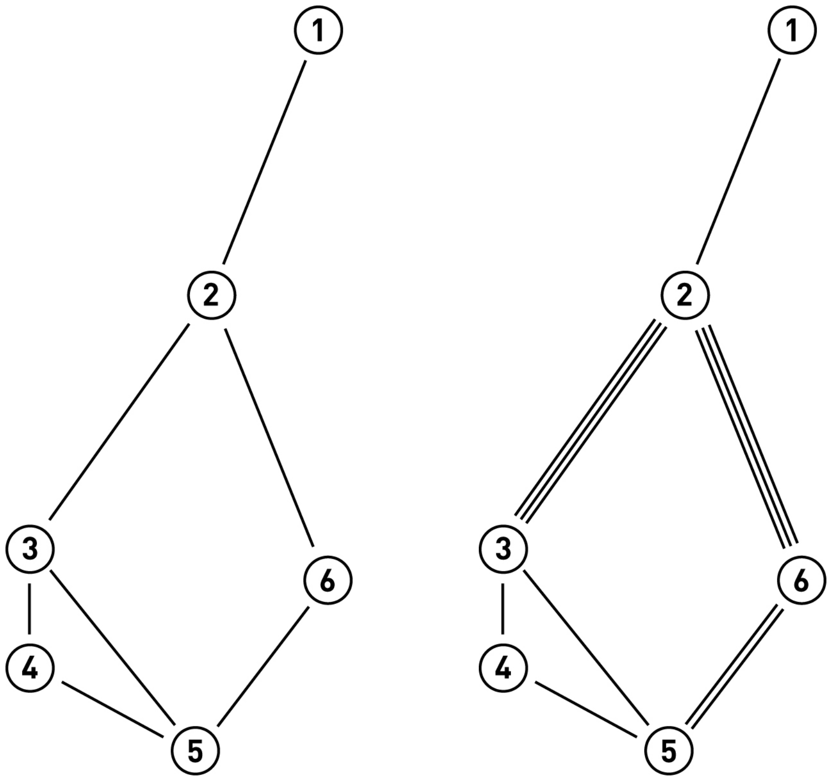

Figure 13: Test 1 Topology - Test load balancing over multiple links.

6.2.2 Load Balancing

Load balancing is, along with connections, configured at each node during configuration of the COOS routers. As the load balancer is defined as a post-processor, a packet will be directed to a channels transpoter from the load balancer. As the load balancers are based on initial configuration, a dynamic calculation is not supported. Since routing matrices are calculated for different QoS parameters it would be natrual to also have load balancers that support this.

6.3 Simulation Runs

A total of over 70 simulations were run through my research. As pointed out earlier, my focus has been the delay through the network and how to minimize it and the packet loss due to timeouts. As explained earlier, the routers tend to give high delays when the number of simultaneous packets being processed increases.

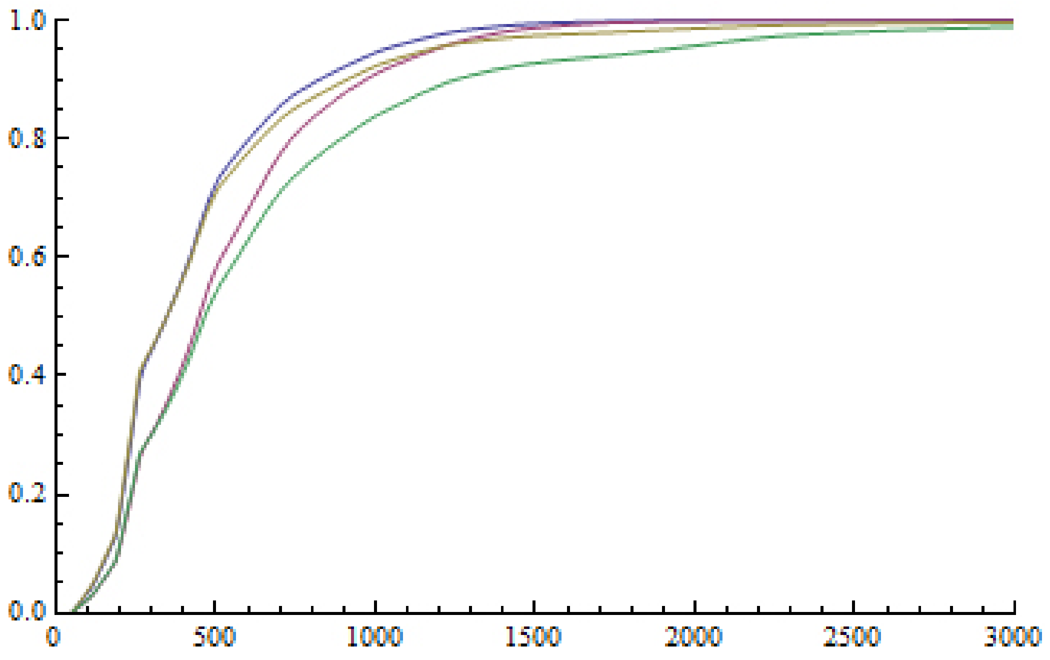

Figure 14: Delay for Type 2 traffic for a Round Robin setting on topology 2 and a normal in topology 1.

Figure 14 shows us the cumulative distribution in delay for two scenarios with a low traffic intensity. One using no load balancing. The other scenario is the same traffic amount, only run with Round Robin on the same topology. Both are illustrated in Figure 13.

One can clearly see from the figure that there is a difference, both between the traffic types and the topologies. The two curves starting off steepest illustrates the higher prioritized traffic. It has an average on about 425 ms. in both cases. The variation on the average between the four different policies tested, including no load balancing, the difference in mean value is only 1,4ms for the *Type 2* traffic (2,3 ms. for *Type 3* traffic).

There is also a difference between the different traffic types in the figure. The average of *Type 2* traffic is 425 ms. in both cases where it is 519 and 521 ms. for the *Type 1* traffic. At this network utilization a 22% growth in delay is alot. Especially when, as pointed out earlier, the most of delay through the routers are paralell and takes no regard to priority. In this case however the difference is due to the difference in number of average hops each packet does. Where *Type 2* traffic used an average of $1,56$, *Type 3* used $1,92$ giving both types of traffic an average of about 270 ms. used pr. hop through the

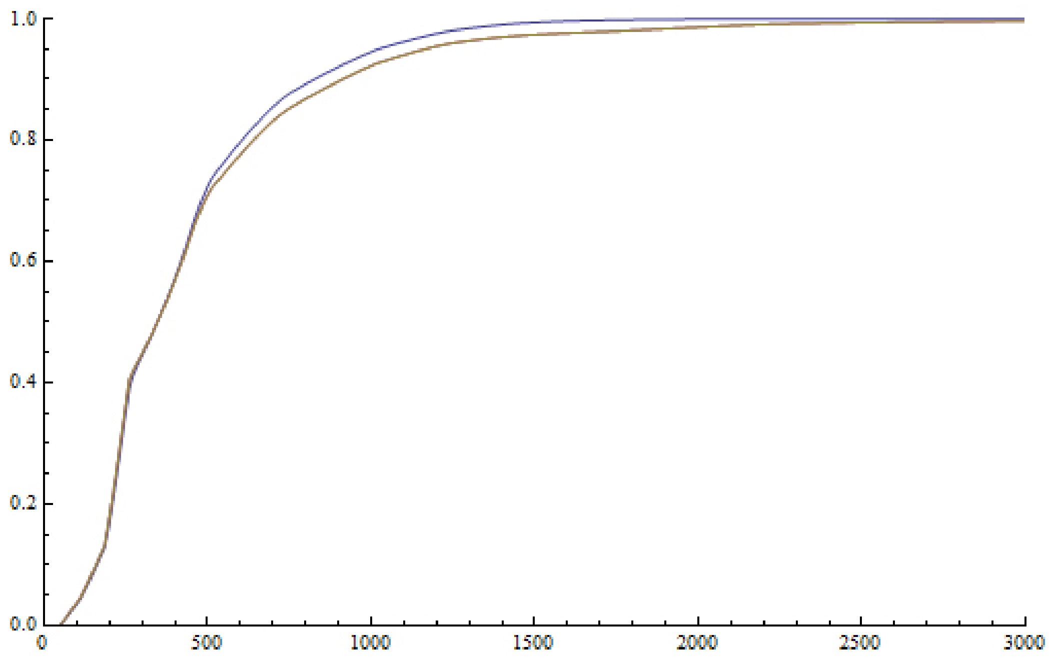

Figure 15: Cumulative distribution of delay for prioritized packets when traffic increases with 0%, 33% and 55% running without load balancing. The two first have identical distribution.

network. This difference is only a result of the traffic pattern defined from the case scenarios.

When we increase the saturation of the network the behavior stays the same. As Figure 15 shows, we need to apply a certain amount of traffic before it affects the delay times noticeably. The delay time increases slowly up against saturation where an exponential increase is experiences. This typical for server/queue-systems and is explained in [13] and [14]. This is also seen regardless of both priority variations and load balancing policy. The lower priority traffic is earlier affected as the queues in the *Transporters* increase.

6.3.1 Adding load balancing nodes

In a utilized information system, adding resources where its needed is a way to increase performance. In my research the router nodes are the focus area as links is an uncontrollable resource. My interest was to see if adding extra routing nodes at points in the topology in the network, load balancing certain traffic between these, would help the performance. As Figure 16 shows I

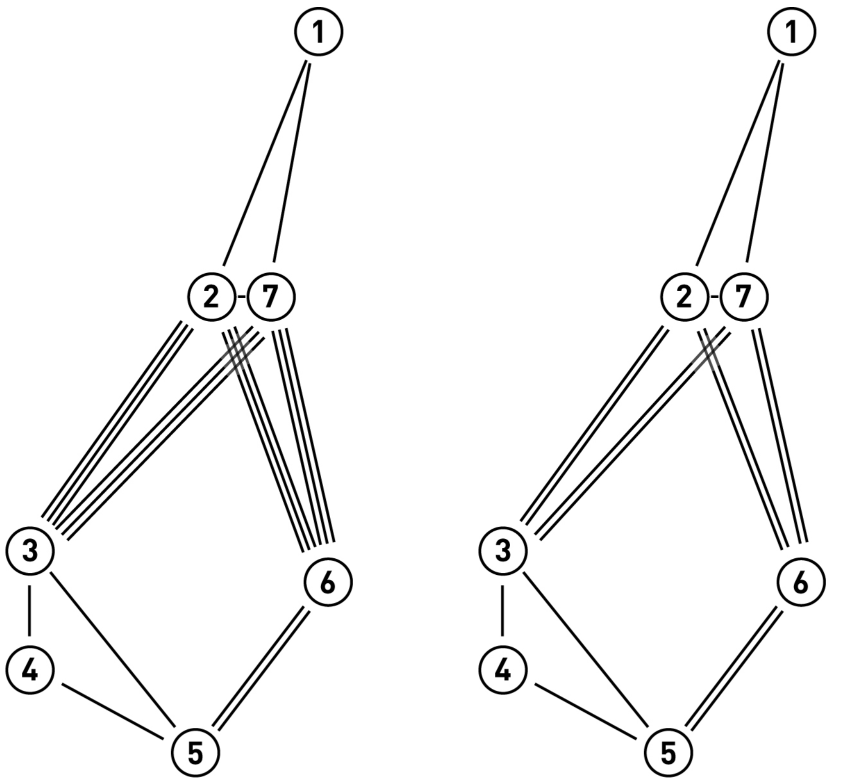

Figure 16: Test 2 Topology - Simulation to test the effect of adding additional nodes.

added a parallel node to node two. All traffic going from node one is then load balanced between these two as it earlier had only one option, regardless of the cost matrix. In addition, there link load balancing is used where the number of additional links added is changed.

6.3.2 Increasing mesh level

By increasing the mesh level of the network, my assumption was that the average hop count of packets will be reduced. This will both decrease the utilization of the routers and reduce the time a packet uses traversing between network layers.

As expected, the hop count decreases drastically, closing in on one. Because of link failures, it does not reach it completely. Simulations were run with a greatly increased link failure rate, one link failure every two hours, in addition to 55% higher load on the network. Even then, a full mesh topology without any extra links outperformed the earlier simulated scenarios. The average hop count was decreased from around 1,56, to around 1,47. This litle decrease resulted in a delay performance on up against 8% for prioritized traffic and 15% on none prioritized traffic. The loss rate is allmost, even with the high link failure rate, allmost nullified. The only packet loss' are isolated events due to a small statistical possibility with a large router delay.

7 CONCLUSION

Load balancing within the network, has to some degree an effect. The extra links through the network creates redundancy of links, which again will give the topology an extra robustness. Links through the IP network is however often in the same Shared Link Risk Group [10], which will decrease this effect. Performance wise for delay through the network, the effect with extra channels, makes the system more robust for higher traffic load. It is however, not been researched if the resources of the system running a router node fully will benefit from the increased channels.

The results also give a clear indication that a key number to strive for in this topology is the hop count a packet uses through the network. As we see in figure 18 a packet traversing through the COOS network, must for each hop “go up” into the COOS layer to be routed to the next hop. With an increased mesh degree and a lowered average hop count, this extra time

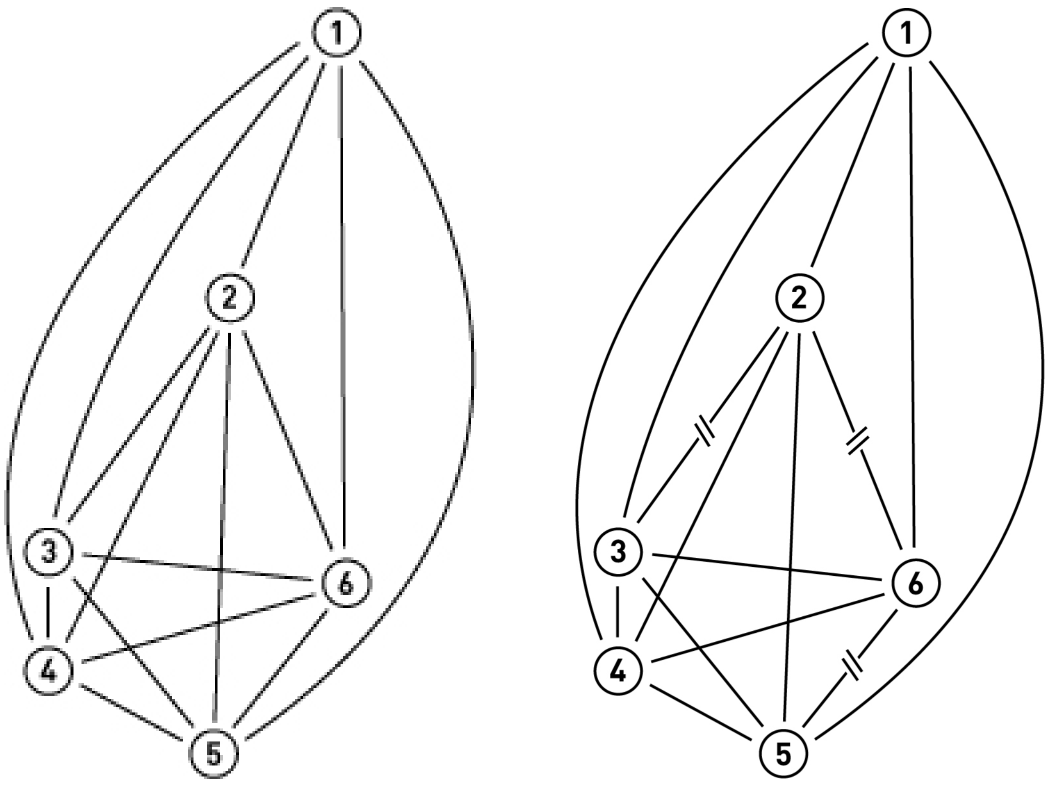

Figure 17: Test 3 Topology - Simulation to test the effect increasing the mesh-level of the network.

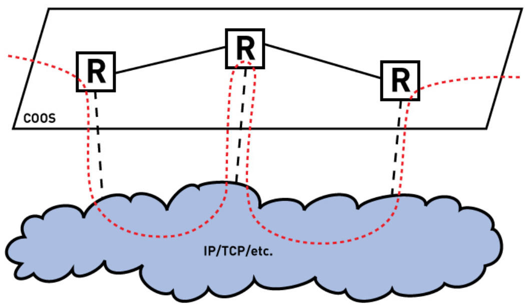

Figure 18: Each packet has to traverse through the network layers for each hop through the network. This increases load on each COOS router which again will affect the performance and experienced QoS.

used can be decreased easily. The increased mesh will also, as we have seen, increase the robustness of the network. Even with rapid link failures, the only noticeable effect is on traffic without a guaranteed delivery.

A full mesh will of course increase the cost on configuration of the routers for every extra link. Firewalls throughout the network can also create problems in reaching a full mesh. The cost of adding extra links does however greatly increase the performance if it is compared to adding extra nodes or configuring several parallel links.

7.1 Future Work

As pointed out, there are several uncertainties on the behavior of the COOS elements. The performance of the routers is only tested as a whole. There are several issues within a router this complex that can greatly affect the performance. Both of the router, but also the QoS experienced as a whole. This counts for starters the performance variation during packet size variation, which we have seen will affect which load balancing policy to use. The use of priority flags throughout the router can also give stricter QoS guarantee for traffic with strict requirements.

A more thorough consciousness-raising of the network aspect of this kind of solution should be done. The traffic pattern expected will affect, both the topology, but also placement of gateways and service platforms. This pattern can also give a wider understanding of how to effectively balance and forward traffic.

References

[1] *Simula webpage*, `http://www.iro.umontreal.ca/~simula/`.

[2] Jan A. Audestad, *Internet as a multiple graph structure*, Information Security Technical Report, No. 1 (2007).

[3] Jon C.R. Bennett and Hui Zhang, *Wf2q: Worst-case fair weighted fair queueing*, `http://lion.cs.uiuc.edu/courses/cs497hou/pres/WF2Q.pdf`.

[4] Graham M. Birtwistle, *Demos: a system for discrete event modelling on simula*, Springer-Verlag New York, Inc., New York, NY, USA, 1987.

[5] Holger Boche, Marcin Wiczanowski, and Slawomir Stanczak, *Unifying view on min-max fairness, max-min fairness, and utility optimization in cellular networks*, EURASIP Journal on Wireless Communications and Networking **2007** (2007), ID 34869.

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to algorithms - 2nd ed.*, MIT Press, 2001.

[7] Wikipedia The Free Encyclopedia, *Posix*, `http://en.wikipedia.org/w/index.php?title=POSIX&oldid=300336312`.

[8] Kevin Fall, *Lecture: Qos in atm*, `http://www.cs.berkeley.edu/~kfall/EE122/lec27/`.

[9] Institute for Computer Science and University Duisburg-Essen Business Information Systems, *Javademos*, `http://sysmod.icb.uni-due.de/index.php?id=javademos`.

[10] Bjarne E. Helvik, *Dependable computing systems and communication networks*, Tapir Akademiske Forlag, 2007.

[11] University in South California Information Science Intstitute, *Network simulator 2*, `http://nsnam.isi.edu/nsnam/index.php/User_Information`.

[12] James F. Kurose, Keith W. Ross, and Bhojan Anand, *Computer networking*, forth ed., Greg Tobin, 2008.

[13] David J. Lilja, *Measuring computer performance: A practitioner's guide*, Cambidge University Press, 2000.

[14] Daniel A. Menascé, Virgilio A.F. Almeida, and Lawrence W. Dowdy, *Performence by design*, Pearson Education, 2004.

[15] Anders Olsson, *Understanding changing telecommunications, building a successful telecom buisness*, John Wiley & Sons Ltd., 2003.

[16] Abhay Kumar J. Parekh, *A generalized processor sharing approach to flow control in integrated services networks*, Tech. report, Massachusetts Institute of Technology, 1992.

[17] Harbor Research, *2009 - 2013 pervasive internet & smart services market forecast*, `http://www.harborresearch.com/HarborContent/reports.html`, 2008.

[18] William Stallings, *Operating systems, internals and design principles*, 6th ed., 2009.

[19] Dimitrios Stiliadis and Anujan Varma, *Latency-rate servers: A general model for analysis of traffic scheduling algorithms*, `http://ieeexplore.ieee.org/iel2/3539/10628/00497884.pdf?arnumber=497884`.

[20] Wikipedia, *Autonomous system — wikipedia, the free encyclopedia*, `http://en.wikipedia.org/w/index.php?title=Autonomous_system_(Internet)&oldid=267450890`, 2008, [Online; accessed 5-February-2009].

[21] Zeromq, *Measuring messaging performance*, `http://www.zeromq.org/whitepapers:measuring-performance`.