COMP591-11C
Final Report

Route Protocol Monitor

Simon Wadsworth BSc

Supervisor: Dr Richard Nelson

©2011 Simon Wadsworth BSc
Abstract

Network monitoring is important because when things go wrong, network administrators need to react and attempt to fix the problem. Current network monitoring and management tools typically do not make use of routing information in a network. By using the routing information flowing through a network a complete topology map can be developed. The project aims to develop a tool to monitor a routing protocol on a network, and uses this new source of network status data to provide updates when the topology of a network changes.
Acknowledgements

I would like to thank Richard Nelson for supervising my project, and giving me helpful advice, and discussions about issues I was facing. I would also like to thank Shane Alcock for performing a code review, and assisting me with some issues that I was having related to the C programming language, and CPU and memory usage. Finally I thank the WAND Group, especially Richard Nelson, Tony McGregor, and Jamie Curtis for their advice and suggestions for improvements to my Conference Presentation, and to Jamie Curtis for providing me with a OSPF trace file from RuralLink.
Contents

1 Introduction 5

2 Background 8
   2.1 Open Shortest Path First ............................... 8
   2.2 Related Work ............................................ 11
       2.2.1 Network Management Systems ..................... 11
       2.2.2 OSPF Monitors ................................. 13

3 Implementation 14
   3.1 Design and Monitor Positioning ......................... 14
   3.2 Libtrace ................................................. 15
   3.3 Decoding OSPF and Detecting Changes ................... 16
   3.4 Data Storage ............................................. 22
   3.5 User Interface ........................................... 22

4 Results 24
   4.1 Small Scale Testing ..................................... 24
   4.2 Large Scale Testing and Scalability ................... 26

5 Conclusion and Discussion 28
   5.1 Future Work ............................................. 29

References 31
Chapter 1

Introduction

Networks can often become very large and complex, and it becomes impractical to use statically configured routes on machines. To overcome this, Network Administrators make use of Routing Protocols, to alleviate the issues associated with static routing. These routing protocols are able to determine the best route through a network for traffic, and can also have quick fail over when an issue occurs.

Network monitoring is a vital area in the management of networks, as network administrators need to know when something has gone wrong, so that they may react to the issue and attempt to fix it. Current network monitoring systems typically require the use of probes (e.g. ICMP Ping probes). However when a routing protocol has been implemented on a network, it is possible that we may not detect an issue that may arise, as the network may simply adjust and route around the problem, meaning that the network monitoring system would still see the link as up, even though the physical link is down, and being rerouted through the network. This is shown in Figure 1.1, where the link between A and B has gone down, but traffic is being sent through other machines, therefore still being received at the destination.

There are several routing protocols in use, with the Border Gateway Protocol, which is used for external internet routing, and IS-IS and Open Shortest Path First, being the main internal network routing protocols now. Open Shortest Path First and IS-IS are Link-State protocols, which contrasts to
Border Gateway Protocol which is a path-vector protocol. Open Shortest Path First is the most common internal routing protocol, with IS-IS being mainly implemented in large telecommunication companies. Open Shortest Path First, is the routing protocol chosen for this project. Open Shortest Path First can originate routes from other routing protocols, and therefore it is possible to determine if a link that is visible in another routing protocol has gone down. This is especially useful for an organisation that may have Border Gateway Protocol speaking routers, which may lose connectivity to outside networks, and therefore alerts need to be generated.

The project aimed to develop a monitoring system that makes use of routing protocol information, that is not commonly used in current systems. The project involved developing a routing protocol monitor that made use of a Link-State protocol to determine the state of links within a network quickly and efficiently, and changes in topology, with minimal overhead, as well as generating a complete map of the network. This report details the achievements and outcomes of the project and research undertaken into developing
such a system. It is divided into several sections, background, implementation, results, and discussion.
Chapter 2

Background

2.1 Open Shortest Path First

Open Shortest Path First is a routing protocol used in enterprise IP networks, and most router vendors now support it, and it has become the main interior gateway protocol (Tanenbaum, 2003). An explanation of the operation of OSPF is included here, however if further information is desired, please refer to RFC2328.

![Network Map](Image)

Open Shortest Path First is a link-state protocol. Every router in a network running Open Shortest Path First has a complete database of the

\(^1\text{Internet Protocol}\)
network, including routers, interfaces and links. Assuming a network layout as shown in Figure 2.3, when OSPF starts up, every router joins a Multicast group on each link that it has. The routers then make use of HELLO Packets. These allow packets are sent out onto the links that the router has Open Shortest Path First configured on. If a router at the other end, understands OSPF then it will reply with a HELLO packet. These packets are used in setting up the link-state adjacencies between the two routers. They are also used in keeping the connection alive, and act as a heartbeat as such. The absence of a HELLO packet is an indication that the link or router is dead. The HELLO packet also carries vital information, such as the time between each HELLO packet, options, the Designated and Backup Designated router, and Router Dead Interval. They can therefore be used to detect links breaking, where the links are not direct connections between the two routers as shown in Figure 2.2

Figure 2.2: Two routers connected by two switches, with the link between the two switches broken

In the past, when Ethernet was a broadcast medium, the Designated, and Backup Designated routers were more important. These routers would originate a Network Link-State Advertisement, that would describe all of the other routers within the network. However in the modern day, most Ethernet networks are point-to-point, and only two routers are classed as on the network, and so therefore Network Link-State Advertisements are likely to only include two routers, the Designated and Backup Designated Routers.

Once OSPF has established a connection and formed adjacencies, the routers exchange Database Description Packets. These packets contain the headers for all Link-State Advertisements that these routers have. The routers then compare the database of it’s neighbors with the database it currently has. It then requests from the neighbor, Link-State Advertisements
that it does not have at present, or that are older than the one described in the Database Description packet. This ensures that when a new router is added, or a new adjacency formed, the routers have the most up to date information in their database. This process of Database exchange is more efficient than sending the entire database across the network (J. T. Moy, 1998).

OSPF uses a method known as reliable flooding, for flooding link changes through the network, which uses IP Multicast. This works by broadcasting link changes over a multicast link, allowing packets to be received by all other routers on that link (J. Moy, 1998). When a router changes its link information, it sends out a Link-State Update packet to all of its adjacencies, which contains Link-State Advertisement for every link that the router currently has. These can be Router, Network, Summary, ASBR summary, and External Link-State Advertisements. OSPFv2 supports other types of Link-State Advertisements, but these are not in common use, and as of OSPFv3 most are deprecated and no longer used at all.

Based on the network layout displayed in Figure 2.3, the following is an explanation of how Open Shortest Path First reacts to topology changes.

Figure 2.3: Network Map showing changes and updates to the topology (Green lines are Link-State Updates)
This assumes that the network has been running for some time, and is stable. However if for some reason, such as a cable being disconnected or a router crashing, and the link between 10.1.1.4 and 10.1.1.3 goes down then router 10.1.1.4 will send a Link-State Update packet to 10.1.1.5 and 10.1.1.2 informing them of the links it contains. This Link-State Update will now no longer contain the Link-State Advertisements for the link to 10.1.1.3. 10.1.1.3 will also send a Link-State Update to 10.1.1.2, which will no longer contain the Link-State Advertisements for the link to 10.1.1.4.

10.1.1.5 and 10.1.1.2 will then forward the Link-State Update packets onto 10.1.1.1. From here it is likely that 10.1.1.1 may forward the Link-State Updates onto 10.1.1.5 or 10.1.1.2 again, however these two routers would ignore it, and not forward it anymore as it has already been received. 10.1.1.2 would also forward the Link-State Update from 10.1.1.4 to 10.1.1.3 and from 10.1.1.3 to 10.1.1.4.

When these routers receive the Link-State Update packets, they will compare their databases to the Link-State Update, and update the database appropriately. This is the part of OSPF that is known as reliable flooding, as previously mentioned. At this point, the routers now have a complete up-to-date database.

All routers on the network will now run Shortest Path Algorithms on their databases. This defines a routing topology that is unique to each router. This means that when a link goes down, Open Shortest Path first will quickly react and modify its routing policies to route around an issue, if there is alternative paths. Because each router participating in OSPF has a complete database map, it has the unique feature, where every router in the network, may build a routing table for any other router (J. T. Moy, 1998).

2.2 Related Work

2.2.1 Network Management Systems

There are several network monitoring packages available, but the two main systems for detecting link outages are SNMP protocols monitors and alerting
systems like Nagios.

There are several systems currently available for use in network management and monitoring. One such system is Simple Network Management Protocol (SNMP). SNMP is an internet standard protocol, for managing devices, that include routers, switches, servers, workstations, printers, modem racks (Mauro & Schmidt, 2005). SNMP works by probing the device for information regarding stored on the device, that provide information regarding the state of the device. There are several SNMP monitors available, such as Cacti and MRTG that make us of the SNMP protocol to receive information from devices on the network. SNMP monitoring of link information requires probing the device constantly, and also requires a connection to the device. Not only this you have to poll each device individually. However by monitoring the routing, it is possible to monitor from one single point in the network, and requires no active traffic polling and management. An advantage SNMP has over monitoring the routing, is that SNMP can provide extra information that routing can not. While not relative to network monitoring, it can be useful in situations to use a system that can provide this information such as temperature of a device.

Another example of a network management system is Nagios. Nagios is an open source network monitoring system. It works by watching hosts and service and providing alerts when things go wrong. It has a simple plugin design that allows for easily extension, and as such it could be possible to write extensions that interface with a route protocol monitor. This again has some disadvantages compared to monitoring a routing protocol. The main disadvantage is that monitoring systems like this must, as with SNMP, poll the devices for information and check they respond to detect link outages. This introduces unnecessary traffic into the network. However an advantage of management systems like this, is similar to the advantage of SNMP. It is possible to discover information about services as well, such as web servers that may crash. However if this is not required then running a fully fledged system like this could be too much overhead.

The biggest advantage that routing monitoring has over other systems, is that it can detect link changes where a routing protocol has hidden the
changes as mentioned earlier.

### 2.2.2 OSPF Monitors

PacketStorm provide Route Analyzer which is a closed source proprietary OSPF analyzer, which provides for network management. It boasts real time monitoring, dynamic data graphs, OSPF recording, and support for 1 GigE, 10GigE and OC-192C (PacketStorm Communications, Inc., 2011). Libtrace supports these network types, and therefore as such, the monitor developed in the project will also.

Shaikh and Greenburg published a paper on the architecture, design and deployment of an OSPF monitor. They developed system that provides real-time tracking of OSPF behavior. As a by product of their research, the system was able to be used to identify problems in the network, however this was not the only purpose for their system. The system also included functions that facilitated off-line analysis of the behaviour of OSPF, including the identification of anomaly signatures, and tuning OSPF configurations (Shaikh & Greenberg, 2004). Their monitor works by directly connecting into the OSPF session, and directly receives LSAs, which is in contrast to the approach taken, discussed in section 3.
Chapter 3

Implementation

3.1 Design and Monitor Positioning

An important problem that needed to be considered is where the monitor would be positioned, how packets would be received and how this would effect the design. Because OSPF makes use of IP Multicast, it would be been possible to develop the monitor in a way, that it used a raw networking socket, and would join a multicast group that OSPF is using. The monitor would then receive all the information that was broadcast by the OSPF routers. However, not all of the packets are multicast, and therefore this method was not actually guaranteed to receive all the Link-State Packets. Ultimately it was decided that it was better to implement the monitor in a completely passive nature, with the only active traffic injection occuring when an OSPF implementation is having to be ran, as in the second scenario described below.

There are several positions that the monitor could be placed passively. This first is where a monitor is connected to a SPAN port\(^1\). Therefore by mirroring a port that is carrying OSPF traffic, it is possible to get a copy of this traffic and process it. This is one method of ensuring the monitor will receive all Link-State Updates required to detect changes.

Another method, is where a monitor is directly connected to a router. If

\(^1\)A port that is configured to mirror the traffic of another port
we are to assume this monitor is on a generic PC, the it is almost impossible to natively capture OSPF traffic, without the monitor speaking the full OSPF protocol. This problem can be solved by running a software package that provides the OSPF implementation, such as the well known open source routing suite Quagga. Quagga can be configured to form an adjacency with a router, and announce it’s loopback address. Therefore the machine will receive all the Link-State Update packets from the router.

Another possible solution is that if you are running some routers, connected to a hub, that is broadcasting the traffic to all connected devices, then you can passively monitor that link. This is due to the fact that the traffic destined for another router, will be broadcast to all other routers, but by switching the NIC into promiscuous mode.

### 3.2 Libtrace

To facilitate the capture and processing of network routing information, the Libtrace library, a C library developed by the WAND Network Research Group at the University of Waikato, and designed for the processing of network traces, was used. Libtrace is an expansive library, that handles a large proportion of accessing the information in a network packet. However Libtrace does not support Open Shortest Path First decoding. Instead this Open Shortest Path First decoding must be implemented separately. The use of Libtrace provides a large benefit. Open Shortest Path First runs over IP, and therefore resides above the IP header in the networking protocol stack as shown in Figure 3.1. Libtrace can decode traffic where there are a large number of varying headers below the Open Shortest Path First headers and content. This therefore provides the ability to deploy a monitor on a range of different network types, and have it work as intended to, by allowing the developer using Libtrace to simply get the IP contents. The main limitation with this, occurs in the support for multiple protocol types that Open Shortest Path First provides.

---

2A mode in NICs that allow a machine to access packets not destined for itself
The choice was made that it would be unwise to implement the entire Open Shortest Path First standard in Libtrace as it is large and complex, and a large proportion of the standard is not needed for monitoring. Therefore not all of the OSPF standard can be decoded by the code that has been written, only the parts required to capture the information for monitoring. It is also worth mentioning that the code written for the Open Shortest Path First decode has not been integrated back into the main Libtrace package yet, and therefore is not currently available for public use.

### 3.3 Decoding OSPF and Detecting Changes

OSPF resides in the packet, above the IP layer. As this is described in section 3.2, it shall not be discussed again. However it is necessary to discuss what parts of the OSPF packets were used, and how these are used. The Hello, Database Description, Link State Request, Link State Update and Link State Acknowledgement packets are used by OSPF for link-state database synchronisation (J. T. Moy, 1998). However for the purposes of monitoring, not all of these packets provide useful information. The Hello packet, which acts like a heartbeat, only provides valuable information to the neighbors of routers. The monitor will not see these, as they are only exchanged between neighbors, so they can not be used to determine the connectivity of links further away. The Link-State Request and Link-State Acknowledgement packets, also provide no valuable information for the monitor. The Link-State request packet only contains information about what one router is asking another router
for, and contains no link information. Similarly the Link-State Acknowledgement packet is only used to let a router know that the information has been received. This is necessary because OSPF runs on IP which is stateless, and has no provision for retransmission. Therefore OSPF needs to handle this itself. The Database Description Packets provide information only during neighbor establishment, and as with the Hello packet are only exchanged between neighbors. These contain Link-State Advertisement headers, but do not contain all of the information. If the monitor was to observe these packets it would then be necessary to request the full Link-State Advertisement, and therefore the monitor would need to implement far more of the OSPF standard.

Lastly there is the Link-State Update packet. This packet can vary greatly in size, but contains the full Link-State Advertisements. As Link-State Update packets are flooded through the network, and monitoring a network for OSPF packets that are flooded out is the superior way to detect OSPF changes (Shaikh, Goyal, Greenberg, Rajan, & Ramakrishnan, 2002), the Link-State Update packets will provide the most information, and all Link-State Updates will be seen by the monitor. For the purposes of the monitor, only the External, Router and Network Link-State advertisements are used. This has the effect of only allowing a single OSPF area to be monitored, however, the information provided from other areas is only a summary, and therefore it is impossible to be able to calculate links between routers from this information.

The first part of the packet to be handled is the OSPF Header as shown in Figure 3.2. This part of the packet is a fixed size, so jumping over the packet is a matter of simple arithmetic. Even though most of the information in the header is ignored, some parts of it provide useful information. The first of these is the OSPF version. This will be present in all versions of OSPF. This allows to check that the packet is indeed an OSPFv2 packet, and throw away the OSPFv3 packet. If the monitor supported both this would be needed to determine what type of processing was needed. The header also contains the packet type, allowing detection of whether the packet is a Hello, Database Description, Link-State Update, Request or Acknowledgement packet. Also
the header contains the total length of the OSPF packet. This is useful to ensure that when reading in the packet, we have not exceeded the length of that packet, and don’t end up reading garbage.

Some packets, such as Hello, are of a fixed size. These are relatively simple to decode, as it requires no extra checking to make sure you haven’t gone over the length of the packet. However, these will not be discussed in detail, as they are not used for the purpose of monitoring. The Link-State Update packet however, can vary greatly in length. Not just because of the varying number of Link-State Advertisements included in the Link-State Update packet, but also due to the varying length of each Link-State Advertisement. This makes the processing of each packet relatively complex. There are some issues can arise, where the length is not a length in bytes, but a count, and therefore some arithmetic must be performed to calculate the number of bytes read out of the total packet length. While this arithmetic is not overly complex, it does require a lot of different cases, and therefore causes the code complexity to increase.

Figure 3.2: OSPF Header Format
(J. Moy, 1998)
Figure 3.3: OSPF Router Link-State Advertisement Format

Figure 3.4: OSPF Router Link Format

Figure 3.5: OSPF Network Link-State Advertisement Format
Link-State Advertisements have a large part of the packet in common. This is because of the Link-State Advertisement header which precedes the actual Advertisement. This is obvious in Figure 3.5.

The Router Link-State Advertisement provides several pieces of important information. It lists all the links that are known about by that router. As a result of this, this packet in particular can vary in length quite greatly. A fair bit of the information in the packets are not needed, and therefore are not used by the monitor. The important information from the Router Link-State Advertisement is the Advertising Router, which is the ID of the router that originated the Router Link-State Advertisement. Then the next lot of information come from the link format that follows the Router Link-State Advertisement packets. The information of use from these is the Link ID. This is the interface address or link address. These provide the list of links that the router identified by the Advertising Router currently has. By comparing these, and finding common links between routers, it is possible to determine the connected routers.

The Network Link-State Advertisement provides information regarding routers that are connected together on the same subnet. The Network Link-
State Advertisement puts different information in positions compared to the Router Link-State Advertisement. In this case the Advertising Router is the router ID of the Designated router on that network. It is important to remember, as discussed in Section 2.1, that in modern day network a Network Link-State Advertisement usually only contains two routers. The Network Mask is also used to determine the network that these two networks are on. This can be applied to the ID to determine this. Finally there are a list of Attached Routers. This will list all the routers on that network, that are connected together. In most common situations there would only be two, but it is possible for there to be many more.

Finally the External Link-State Advertisement differs quite a bit from the others. It is generally one size, although can in some cases extend to different lengths. There is several important pieces of information in these packets. The Advertising Router as with the other Link-State Advertisement contains the ID of the router that originates the Link-State Advertisement. The Link State ID is the prefix of the external route that is imported. The network mask is the subnet of the prefix. So for example, the Link State ID may contain 203.114.153.0 and the Network Mask may contain 255.255.255.0. The External route would therefore mean that the addresses in the range of 203.114.153.0 to 203.114.153.255 were accessible through the router identified by router ID.

Because all of the routers on the network, will send a complete list of the Link-State Advertisements in their Link-State Update packets, it is possible to notice if a Link-State Advertisement is now missing. This therefore makes it extremely easy to detect if something has changed in the network, due to information missing. This can also apply to external routers, even though it is not possible to detect what has actually gone wrong. What it is possible to detect however, is that something has gone wrong, as is indicated by the external router disappearing.
3.4 Data Storage

Another important decision needed in the implementation, is how to keep track of the data. There were two obvious solutions, storing the data in a data structure within the program, however this is volatile and is lost when the program exits. Or storing the information in some sort of database system, such as MySQL or PostgreSQL. However database systems like these are bulky and complex, and for the purpose of a single monitor are not required. Therefore it is practical to use a system like SQLite which provides several advantages. The first of which is that the SQLite code can be easily compiled and linked into the application. This provides the advantage of the monitor not needing any extra database software installed on the system to which it is to be deployed. A second advantage, one that is not solved just by SQLite however, is that easy querying of the current link database is possible. This allows the ability to check if a router is already in the database, and provide quick and easy insertion and updating of the database. Further is makes the testing of link information, and finding of connected routers easy, by using SQL queries to find routers which have common interfaces. By keeping flags in the database surrounding the validity of interfaces, and also of the time that interface/route was last seen, it is possible to easily detect if things are expired, or no longer invalid.

3.5 User Interface

To enable the user of the monitor to be able to determine if something has gone wrong in the network, some type of interface was needed. There were several considerations related to the interface, regarding what should be displayed and in what manner. It was decided that it would be most appropriate to develop a web based User Interface, that could therefore be accessed from anywhere on the network. Due to the web-based natured desired, the PHP programming language was used to develop and implement the interface. The interface displays a list of routers that are detected. If the network has a Domain Name Server in it, with Router IDs mapping to Domain Names, the
monitor will display these. This is to allow the easy identification of routers, especially in networks where they have a specific naming scheme. The user is then able to select a router, and view information regarding links, interface and external routes. These are colour coded, Red, Blue or Black. Red is used to indicate that a route is invalid, Blue that it has expired, but could still be valid, and if everything is fine it will remain in Black.

<table>
<thead>
<tr>
<th>OSPF ROUTE NETWORK MONITOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROUTERS</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>10.2.1.1</td>
</tr>
<tr>
<td>10.2.1.2</td>
</tr>
<tr>
<td>10.2.1.3</td>
</tr>
<tr>
<td>10.2.1.4</td>
</tr>
<tr>
<td>10.2.1.5</td>
</tr>
</tbody>
</table>

Figure 3.7: The Monitor Interface

To enable a network administrator to easily identify and issues that may be present on the network, they can easily get a list of alerts, which displays any changes that may have happened. This will only display things that are no longer valid, or else have expired. To also assist in the monitoring of the network, the interface implements a simple Force Directed Network Map. The JavaScript Infoviz Toolkit was used to develop this, due to its ease of use, and simplicity at creating and displaying Force Directed Graphs. To allow use of the monitor to be displayed without user interaction, the interface will refresh at prespecified intervals.
Chapter 4

Results

4.1 Small Scale Testing

The test the monitor, it was necessary to test it against several different network topologies. This was achieved by using the router operating system, Vyatta, installed on multiple virtual machines. This use of virtualisation, allowed the network to be altered quickly, without needing physical hardware, or the potential for things to go wrong. The network first used for testing used a topology like that in Figure 4.1. For testing, the monitor was deployed on the link between 10.1.1.2 and 10.1.1.1. In relation to Section 3.1 this is equivalent to the connecting it to a hub, as all machines on the link are receiving the packets. The monitor was set to passively monitor the interface, which was running on the same physical network as the link between those two routers.

The monitor was set running, and waited until it established a complete map. It is important to note that this can take some time if the monitor is not started before the routers. This problem would not exist if you were running the monitor in the configuration mode where an OSPF implementation was running on the same machine as the monitor. Once the complete topology was detected by the monitor, configuration changes were made to the network. This included taking interfaces down on routers, disabling links on routers independent of the operating system, and configuring and changing
external routes. These were detected by the monitor, all within the expected time periods. When a interface was brought up or taken down, this was almost instant, as Link-State Update packets were flooded through the network almost instantaneously, and the monitor could pick these up quickly.

When a link was cut, and required the routers to detect this change, it could take up to 30 seconds, depending on the router configurations. This is due to the amount of time required for the Hello timeout to be reached, and Open Shortest Path First to infer the topology change, and announce the new updated list of Link-State Update packets.

Because this network was tightly controlled, and the validity of the links were controlled, it made it very simple to determine if the monitor was correctly detecting changes. The monitor correctly read all of the OSPF Link-
State Update packets, and processed them into the database correctly. After it was determined that his was correct, it was checked that the monitoring interface was displaying the information correctly, which was deemed to be the case. For all links that had gone down, they were coloured Red, and when a link came back up, it was coloured black again.

4.2 Large Scale Testing and Scalability

To test if the system would detect changes correctly on larger networks, it was ran against a 24 hour trace from RuralLink’s network, which comprises of over 100 Open Shortest Path First enabled routers. However RuralLink implements multiple Areas, and as a result not all of these routers were detected, however the monitor did detect all of the default area routers, as this was the area from which the trace was taken. This is the behaviour that is to be expected, as the implementation did not make use of Area Link-State Advertisements, and therefore is not expected to detect the routers outside of the Area. The system handled the traffic load well, due to a feature of OSPF mentioned in the next paragraph.

The program is expected to scale well. Because Open Shortest Path has a minimal traffic footprint, and assuming a network is relatively stable, the amount of traffic being observed by the monitor, will not be large. Therefore it is not expected to need a large amount of resources to maintain the monitor. The CPU usage of the monitor is relatively small, due to the use of events, which cause the application to sleep and wait until a software interrupt occurs, notifying of the arrival of a packet. Also the memory usage is relatively small, as the packet is only kept in memory for as long as it’s needed, and freed afterwards.

The monitor could be ran over the entire 24 hour OSPF trace from RuralLink in less than a minute when done in real time. This shows that the overhead of using OSPF monitoring is rather small, as the total time needed to process the entire trace file was relatively small.

\footnote{Libtrace provides an option for trace time, that would cause processing to take the same length as the trace}
One issue that was discovered with the interface, especially on large networks, is the response time of the Network Map. It can take some time to render and layout the map, however as this was implemented as a nicety, the performance and improvements of displaying this was beyond the scope of the project.
Chapter 5

Conclusion and Discussion

The project aimed to develop a system for monitoring a network for changes, by using routing protocol information as a new source of information. This was desired as some monitoring systems can not detect changes in the topology of the network, as they will still detect a link being up, even if the physical link is down.

Development of code to access OSPF packets was a difficult and interesting learning experience. It required working with raw bytes off the network, that were provided from Libtrace. There were several reasons for this. First was that although the C Programming Language was not new to me, it was the first time I had used it to process network traffic manually. In previous experiences of using C and Libtrace especially, I only had to use information provided from Libtrace itself, which in this case was already the correct byte order, and type. However in the development I had to take into account the types of information that the bytes represented, and also the byte order. There was also issues surrounding common buffers in use by some functions, that cause strange results when used in the same statement. However, this was fixed by moving to a newer version of the function, after discovering the old one was deprecated.

Another difficult problem that was encountered was that of using SQLite in C. It is relatively simple to use SQLite in PHP and extracting information from a database is relatively intuitive. However in C the method of building
and executing queries, and getting information for a table is rather complex and interesting. During this I discovered some neat ways of building queries to execute on SQLite. One problem I did find with SQLite is the lack of table editing. It is almost impossible to alter the fields in a table once it has been created. Instead basically the entire table must be deleted and recreated. This is an important lesson, as it highlights the importance of thinking through the database layout before developing. A lot of time was spent deleting and recreating tables, that could have been avoided if more time was spend on database layout, and thinking about what information would be stored, how it would be linked together, and how it would be displayed.

One of the big things that was learnt was the extraordinary amount of information in OSPF. Further I have learnt a great deal about OSPF, and routing protocols in general, that I have not learnt through taught classes. But further the extraordinary amount of information that can be provided about a network and used for monitoring. While not all of it was used in this project, there is certainly information present in OSPF that could be used to improve the user interface.

In the end, the project has achieved its aims. Through the course of the project, investigation into the OSPF was undertaken, and the possibility of using it to monitor links and topologies within networks. It was determined this was possible, and then code was written that allowed the decoding of OSPF. After this code was written it was updated and reworked several times to address CPU usage and memory issues. The system finally resulted in a C daemon that would monitor a network and store information in a database. From here a user interface was developed that allowed the display of the information, and allows network operators to see any problems in the connectivity of their network.

5.1 Future Work

There are several possibilities for future work. The first is in the area of OSPF Area support. Currently multiple areas are not supported due to
the fact, as mentioned in the Implementation section, that it is not possible to determine the connection of routers outside the area that the monitor is in. However it would be entirely feasible to modify the system to use, for example, MySQL as a distributed database, allowing multiple monitors in each area, feeding information into the database. This would also require modification to the monitor code, to read, process, and differentiate between different OSPF Areas.

Another area for further work is, as this project did not take into account other routing protocols such as IS-IS and OSPFv3, then there is work that could be done to implement other routing protocols along side OSPF, and make the monitor useful in situations where network operators are using multiple routing protocols. This would be specifically useful with OSPFv3 as network operators begin moving to IPv6 deployment, and using the OSPFv3 routing protocol for IPv6. This would require similar work in terms of decoding other routing protocol packets, and storing the information in the database. This would require modifications to the processing code, to allow for the extra types of protocols that may be seen, as the current system has restrictions to only allow OSPF.

Another further area of work, would be to implement support for systems like SNMP or Nagios. Allowing the monitor to send SNMP traps, or integrate with the Nagios Network Monitoring System, would be beneficial to network operators using these systems in their networks. Specifically for Nagios, by adding a plugin to Nagios, that used the information discovered by the monitor, would reduce the need for a separate interface, and allow the Network Operator to deal with a single point of information.
References
