Designing and implementing ethernet networks

Joachim Tingvold

ALICE HLT, CERN

Bergen University College

Abstract

Implementing and designing ethernet networks requires thought to several aspects. One would like to build a stable, scaleable, and fully redundant network, as possible, while keeping downtime to a minimum.

Contents

List of Figures

List of Tables

List of Listings

Glossary

ALICE	ALICE (*A Large Ion Collider Experiment*) is one of the six detector experiments at the Large Hadron Collider at CERN. (1)
AS	Within the Internet, an Autonomous System is a collection of connected Internet Protocol (IP) routing prefixes under the control of one or more network operators that presents a common, clearly defined routing policy to the Internet. (2)
BMC	The baseboard management controller is the intelligence in the IPMI architecture. It is a specialized microcontroller embedded on the motherboard of a computer, generally a server. The BMC manages the interface between system management software and platform hardware. (3)
Broadcast radiation	Broadcast radiation is the accumulation of broadcast and multicast traffic on a computer network. Extreme amounts of broadcast traffic constitute a broadcast storm. A broadcast storm can consume sufficient network resources so as to render the network unable to transport normal traffic. (4)
CARP	The Common Address Redundancy Protocol or CARP is a protocol which allows multiple hosts on the same local network to share a set of IP addresses. Its primary purpose is to provide failover redundancy, especially when used with firewalls and routers. (5) (6)
CERN	The acronym CERN originally stood, in French, for *Conseil Européen pour la Recherche Nucléaire* (European Council for Nuclear Research), which was a provisional council for setting up the laboratory, established by 11 European governments in 1952. The acronym was retained for the new laboratory after the provisional council was dissolved, even though the name changed to the current *Organisation Européenne pour la Recherche Nucléaire* (European Organization for Nuclear Research) in 1954. (7)
CHARM	The CHARM (Computer-Health-And-Remote-Monitoring) card is plugged into the PCI-bus of the host-computer. It is a small single-board computer running Linux as operation system that can monitor and control the host computer. Connections to the CHARM card are possible via its own network for full remote control of the host system. (8) (9)
DHCP	The Dynamic Host Configuration Protocol (DHCP) is an automatic configuration protocol used on IP networks. Computers that are connected to IP networks must be configured before they can communicate with other computers on the network. DHCP allows a computer to be configured automatically, eliminating the need for intervention by a network administrator. It also provides a central database for keeping track of computers that have been connected to the network. This prevents two computers from acciden-

	tally being configured with the same IP address. (10)
DNS	The Domain Name System (DNS) is a hierarchical naming system built on a distributed database for computers, services, or any resource connected to the Internet or a private network. Most importantly, it translates domain names (or hostnames) into the numerical identifiers associated with networking equipment. (11)
EtherLock	EtherLock is a closed loop monitoring system, similar in function to a home alarm system. Just as a home alarm sounds when a door or window is opened, connecting or disconnecting any network asset (workstation, notebook, printer, server, hub/switch) is detected in real time. (12)
Gateway	A gateway is a network point that acts as an entrance to another network. On an IP network, clients should automatically send IP packets with a destination outside a given subnet mask to a network gateway. A gateway is an essential feature of most routers, although other devices (such as any PC or server) can function as a gateway. (13)
GPN	General Purpose Network, a network providing Internet access for computers at CERN.
HLT	High Level Trigger. (8)
InfiniBand	InfiniBand is a switched fabric communications link used in high-performance computing and enterprise data centers. Its features include high throughput, low latency, quality of service and failover, and it is designed to be scalable. The InfiniBand architecture specification defines a connection between processor nodes and high performance I/O nodes such as storage devices. (14)
IP-address	An Internet Protocol address (IP address) is a numerical label assigned to each device (e.g., computer, printer) participating in a computer network that uses the Internet Protocol for communication. (15)
IPMI	The Intelligent Platform Management Interface (IPMI) is a standardized computer system interface used by system administrators to manage a computer system and monitor its operation. (16)
ISC	Internet Systems Consortium, Inc., also known as ISC, is a Delaware-registered, public benefit non-profit corporation dedicated to supporting the infrastructure of the universal connected self-organizing Internet by developing and maintaining core production quality software, protocols, and operations. ISC is the developer of several key Internet technologies that enable the global Internet; such as BIND and ISC DHCP. (17)
MAC-address	A Media Access Control address (MAC address) is a unique identifier assigned to network interfaces for communications on the physical network segment. MAC addresses are used for numerous network technologies and most IEEE 802 network technologies including Ethernet. Logically, MAC addresses are used in the Media Access Control protocol sub-layer of the OSI reference model. (18)
NAT	In computer networking, network address translation (NAT) is the process of modifying IP address information in IP packet headers while in

transit across a traffic routing device. It is common to hide an entire IP address space, usually consisting of private IP addresses, behind a single IP address (or in some cases a small group of IP addresses) in another (usually public) address space. To avoid ambiguity in the handling of returned packets a one-to-many NAT must alter higher level information such as TCP/UDP ports in outgoing communications and must maintain a translation table so that return packets can be correctly translated back. RFC 2663 uses the term NAPT (network address and port translation). Other names for this type of NAT include PAT (port address translation), IP masquerading, NAT Overload and many-to-one NAT. Since this is the most common type of NAT it is often referred to simply as NAT. (19)

Network loop A Switching loop or Bridge loop occurs in computer networks when there is more than one Layer 2 (OSI model) path between two endpoints (e.g. multiple connections between two network switches or two ports on the same switch connected to each other). The loop creates broadcast radiation as broadcasts and multicasts are forwarded by switches out every port, the switch or switches will repeatedly rebroadcast the broadcast messages flooding the network. Since the Layer 2 header does not support a time to live (TTL) value, if a frame is sent into a looped topology, it can loop forever. (20)

Node A node is any device connected to a computer network. Nodes can be computers, personal digital assistants (PDAs), cell phones, or various other network appliances. (21)

OSI model The Open Systems Interconnection model (OSI model) was a product of the Open Systems Interconnection effort at the International Organization for Standardization. It is a way of sub-dividing a communications system into smaller parts called layers. Similar communication functions are grouped into logical layers. A layer provides services to its upper layer while receiving services from the layer below. On each layer, an instance provides service to the instances at the layer above and requests service from the layer below. (22)

OSPF Open Shortest Path First (OSPF) is an adaptive routing protocol for Internet Protocol networks. It uses a link state routing algorithm and falls into the group of interior routing protocols, operating within a single autonomous system. (23)

PF PF (Packet Filter, also written pf) is a BSD licensed stateful packet filter, a central piece of software for firewalling. (5) (24)

Rack unit A rack unit or U (less commonly RU) is a unit of measure used to describe the height of equipment intended for mounting in a rack. One rack unit is 1.75 inches (44.45 mm) high. (25)

Rooted When someone uses a rootkit, or the like, to gain root-access to a Linux/Unix-based OS.

Rootkit A rootkit is software that enables continued privileged access to a computer while actively hiding its presence from administrators by subverting standard operating system functionality or other applications. The

	term rootkit has negative connotations through its association with malware. (26)
STP	The Spanning Tree Protocol (STP) is a network protocol that ensures a loop-free topology for any bridged Ethernet local area network. The basic function of STP is to prevent bridge loops and ensuing broadcast radiation. Spanning tree also allows a network design to include spare (redundant) links to provide automatic backup paths if an active link fails, without the danger of bridge loops, or the need for manual enabling/disabling of these backup links. (27)
Subnet	A subnetwork, or subnet, is a logically visible subdivision of an IP network. The practice of dividing a network into subnetworks is called subnetting. All computers that belong to a subnet are addressed with a common, identical, most-significant bit-group in their IP address. This results in the logical division of an IP address into two fields, a network or routing prefix and the rest field. The rest field is a specific identifier for the computer or the network interface. (28)
Throughput	In communication networks, such as Ethernet or packet radio, throughput or network throughput is the average rate of successful message delivery over a communication channel. This data may be delivered over a physical or logical link, or pass through a certain network node. The throughput is usually measured in bits per second (bit/s or bps), and sometimes in data packets per second or data packets per time slot. (29)
Trunk	Link aggregation or IEEE 802.1AX-2008 is a computer networking term which describes using multiple network cables/ports in parallel to increase the link speed beyond the limits of any one single cable or port, and to increase the redundancy for higher availability. Link aggregation is often abbreviated LAG. Other terms include trunking, link bundling, Ethernet/network/NIC bonding, NIC teaming, port channel, EtherChannel, Multi-link trunking (MLT), Smartgroup (from ZTE), and EtherTrunk (from Huawei). (30)
TTL	Time to live (TTL) is mechanism that limits the lifespan of data in a computer or network. TTL may be implemented as a counter or timestamp attached to or embedded in the data. Once the prescribed event count or timespan has elapsed, data is discarded. In computer networking, TTL prevents a data packet from circulating indefinitely. (31)
VLAN	A virtual local area network, virtual LAN or VLAN, is a group of hosts with a common set of requirements that communicate as if they were attached to the same broadcast domain, regardless of their physical location. A VLAN has the same attributes as a physical local area network (LAN), but it allows for end stations to be grouped together even if they are not located on the same network switch. LAN membership can be configured through software instead of physically relocating devices or connections. (32)
VRRP	Virtual Router Redundancy Protocol (VRRP) is a non-proprietary redundancy protocol described in RFC

3768 designed to increase the availability of the default gateway servicing hosts on the same subnet. This increased reliability is achieved by advertising a virtual router (an abstract representation of master and backup routers acting as a group) as a default gateway to the host(s) instead of one physical router. (33)

1

Summary

The project is about redesigning a ethernet network in a computer-cluster with around 230-250 nodes, at ALICE HLT, which is an experiment at CERN. The first thing we're going to do, is to look at their current setup. We're then going to define the requirements and goals for the new network, before heading on to the actual design. Here we'll go through how the new IP-scheme is going to be, how the network-layout is going to be, and how network-services is going to be implemented. After that, we'll go through the configuration of the switches and servers in the new network. We're also going to do some testing of the new network, to see that it's working as intended.

2

Introduction

My project is about reconstructing a network in a high-performance computing cluster, consisting of about 230-250 nodes. Rip it apart, design it from scratch, and implement it. In this thesis, I'll be talking about the current setup and what the problem is. We'll investigate what requirements the new network has, and how I plan to implement it.

2.1 People

There are a few people worth mentioning, that was involved in this project, one way or the other, both directly and in-directly. In no specific order of preference;

- Hege Erdal (Bergen University College), my advisor.
- Håvard Helstrup (Bergen University College), my team-leader at CERN.
- Olav Smørholm, system administrator at CERN.
- Pierre Zelnicek, system administrator at CERN.
- Øystein Haaland, system administrator at CERN.
- Torsten Alt, technical coordinator at CERN.
- Artur Szostak, physicist/technical coordinator at CERN.
- Camilo Lara, software engineer at CERN.
- Timo Breitner, physicist/software engineer at CERN.

- Timm Steinbeck, software engineer at CERN.
- Jochen Thaeder, system administrator at CERN.
- Volker Lindenstruth, my group-leader at CERN.

2.2 Company

The project has been carried out at CERN - more specifically ALICE HLT, which is where I work. CERN is an international organization whose purpose is to operate the worlds largest particle physics laboratory. It's situated on the border between France and Switzerland, near Geneva. It was established in 1954, and currently has twenty European member states.

2.2.1 ALICE

ALICE is one of the largest experiments in the world devoted to research in the physics of matter at an infinitely small scale.

2.2.2 HLT

HLT is a part of the ALICE-experiment. It combines and processes all the information from all the major detectors of ALICE in a large, high-performance computer cluster. It's task is to select the relevant part of the huge amount of incoming data. This high-performance cluster consists of several hundred servers, which is connected together with two types of network; ethernet and InfiniBand.

2.3 The problem

A lot of the work on, and around, ALICE has been on-going since the 90's. There was a lot of decisions made back then, which unfortunately, in some cases, has been kept that way till this very day. Of course, this is not always a bad thing; if it works, don't fix it. However, some things needs to be regularly maintained and updated. One of these things are computer networks. The fundamental concepts of ethernet networks hasn't changed, but other aspects has. Many of the choices that the current network is based upon, is no longer valid. This is something that I'll discuss in the next section, one of which are the root-cause to one of their biggest problems; IP-exhaustion. Every time they needed available IP-addresses, they had to spend days to find them - literally. The other problem they have, is that there is no real redundancy for critical parts of the network (DHCP, DNS, gateways, etc).

2.4 Current setup

The current setup is really simple. The switches has been configured with an IP-address (for management) on it's default VLAN, and that's pretty much it - so it's literally plug-and-play. Their entire network is the same subnet, 10.162.0.0/16. One should think that 2^{10} (65 536) IP-addresses would be enough for a cluster consisting of around 230-250 nodes. Well, not in this case. The thing is, when the network first was planned way back, it was by someone with electronics-background, who came up with the idea of using bit-flipping. An IPv4-address consists of 32 bits (or 4 bytes), and is often presented in dot-decimal notation, which consists of four decimal numbers, each ranging from 0 to 255, separated by dots - for example 10.162.11.211. Each of the four parts represents a group of 8 bits (1 byte), and is referred to as an octet. The binary representation of 10.162.11.211 would be 00001010.10100020.00001011.11010011. Going back to the bit-flipping, one could clearly see that it could be useful in some cases. The nodes in the HLT-cluster is distributed amongst several rows of racks, over two different floors. What they then did, was to map all the bits to the different rows and racks, so that each two rack unit in the racks, got it's own IP-address. By doing this, you could, by looking at the binary form of an IP-address, know exactly where the node using that IP would be located. Going the other way, if you wanted to insert a new node into a rack, you could easily find out what IP-address it should have, based on it's location in that specific rack.

There was nothing wrong with this design back then - to be honest, it seemed to be a nice system. However, as time changed, they now have servers consuming 2 rack units, but with 4 nodes in it, and each node also has BMC, which requires it's own IP-address. You now suddenly have 4 nodes requiring 8 IP-addresses, whereas you only had 1 node requiring 1 IP-address before, which kind of breaks the entire design. The real problem, though, is that the LDAP-database they use for DHCP and DNS, has pre-reserved all the IP-addresses that was used for the bit-flipping in such a way that it was impossible to determine if an IP-address was in use or not (when they had 1 node per 2 rack units, this wasn't a problem, as they could just follow the bit-flipping system without caring about if it was in use or not). Thus, each time they needed free IP-addresses, they spent loads of time finding free IP-addresses.

The current setup doesn't have any specific redundancy at all (except that some of the switches has trunks consisting of two cables), and all the network-services (including the default gateway) only has some degree of redundancy, of which all has to be done manually. So, if a gateway goes down, the cluster is unreachable externally - you have to physically be present in the cluster, shut down the gateway, and start up the secondary gateway. Not ideal at all.

All the switches are located in one place, and is the cause of a huge cable-mess, which makes it really hard to debug errors – at least if you have to find out where a specific cable is connected. There is also a lot of old mess laying around, such as intermediate home-switches (that is, small 8-port switches) that are interconnected and daisy-chained.

2.5 The company's main goals

ALICE want's a network that is reliable, redundant, and more adaptable for changes and future growth, than what the current setup has.

3

Task definition

Before we can start planning the new design, we need to plan the planning, so to speak. We need to figure out the requirements that ALICE HLT has for the new network, and what conditions applies. We also have limited time, so we need to plan ahead - set milestones with deadlines, which we have to follow. There's also a lot of risks involved, which need to be thought of beforehand, so that we don't do anything stupid. I also want to make a list of our main goals, which is going to be a bit more technical then the requirements and goals that the company has.

3.1 Requirements

The requirements to the new network, as requested from ALICE HLT, was discussed, and agreed upon, during a yearly meeting early december 2010.

- "Automatic" redundancy. There should be no need for human intervention to ensure this redundancy, and should only be required for critical services, such as DHCP, DNS, gateways and core-switches. (Redundancy in other parts of the network and/or system is of course a good thing, but is not a requirement, and should not take the focus away from the critical services)

- Physical restructuring. There are too much cables around the switches, which should be cleaned up. Moving the switches to different racks is no requirement, as long as it becomes easier to troubleshoot cable-errors.

- New IP-scheme. It shouldn't take forever to find free IP-addresses. No specific requirements as to how it's done, as long as it's not the way it is today.
- Separation. A strong wish to physically seperate the two parts of the cluster, production and development, in such a way that they cannot interfere with each other.
- Security. The cluster has been rooted more than once, which they want to avoid, if possible.
- Monitoring. There is a wish for improved monitoring of the network, as this is lacking in the current setup.
- Documentation. Almost no documentation done of the current setup. This should be done for the new setup.

3.2 Conditions

The restructuring should be done by using as much as possible of the current hardware & software we currently have. The restructuring should also be completed in time, or as a last-resort, be able to revert to the old setup quick, as it must be fully operational again by mid-march.

3.3 Hardware & software

We currently have around 230-250 nodes, spread across two floors, CR2 and CR3. Each floor has three lines of racks (X, Y, Z), each having different numbers of racks (from 10 to 17 racks). We only use 4 of the racks in CR3, whilst almost all racks are in use in CR2.

3.3.1 Switches

We have 4 different kinds of switches in the cluster.

3.3.1.1 HP ProCurve 3400cl (J4905A)

We have 1 of these, acting as our GPN-switch. It's a 24-port Gbps-switch (34), with a two-port 10GE fibre-module (35) with one 10GE X2-SC LR transceiver (36).

3.3.1.2 HP ProCurve 3400cl (J4906A)

We have 4 of these, acting as our edge and/or core-switches in the current setup. It's a 48-port Gbps-switch (37).

3.3.1.3 HP ProCurve 3500yl-48G (J8693A)

We have 22 of these, acting as our edge and/or core-switches in the current setup. It's a 48-port Gbps-switch with PoE-support (38).

3.3.1.4 Netgear FSM726 v2

We have 39 of these, acting as our edge-switches for management (IPMI, CHARM, BMC). It's a 26-port switch, where 24 are FastEthernet (100Mbps), and 2 Gbps-uplinks (39).

3.3.2 Software

Current setup uses patched versions of ISCs DHCP and BIND, both with an LDAP-backend. Only halfway redundancy, since secondary servers has to be started manually. LDAP is "redundant" in the way that it's database is dumped with a cronjob, and then manually copied over to the secondary server, imported, and then the secondary server is ready. The gateways is running Ubuntu, using iptables to NAT the public IP to the internal network.

3.4 Purchase

We need to purchase power-cables to the switches, so that we have the opportunity to move them around in the cluster. These power-cables are somewhat special, since the plug that's plugged into the switches, is a C15. Since the racks have C14-outlets, we need C14 to C15 cables. We also need to buy premium license (40) for the core-switches, which will enable us to use OSPF and VRRP.

3.5 Timeframe

Since the cluster is gathering data from the LHC, which is in use almost 24/7, there usually isn't much time to do changes that requires downtime. Doing all the above mentioned changes, isn't something that is done within a weekend, and hence this isn't something we normally can do. However, from the start of december 2010, till mid-march 2011, the LHC is being shut down for a long, technical break, which leaves us with a golden opportunity to make the desired changes. Not all changes has to be done before the LHC starts again (such as monitoring, documentation, etc).

3.5.1 Milestones

I've divided the project into several subtasks, each with it's own milestone and deadline. They are to be completed in chronological order, since they very much depend on each other.

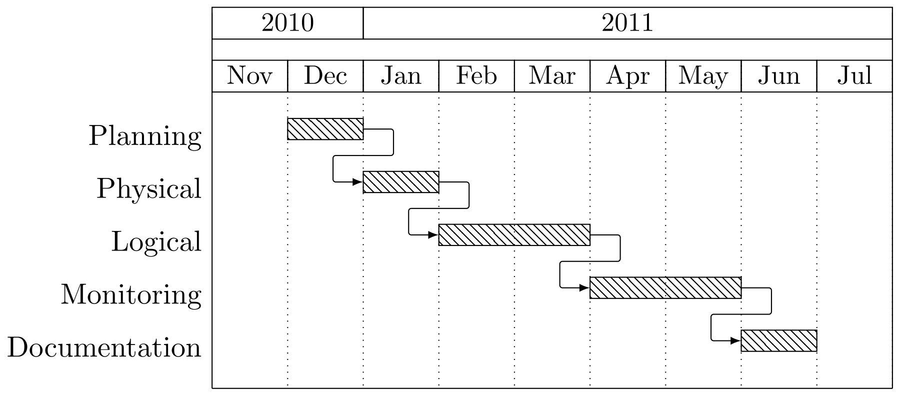

Figure 3.1: Milestones - Gantt-diagram showing the milestones of the project.

3.5.1.1 Planning

By the end of december 2010, all the fundamental planning of the network should be done. This includes the plans of how the physical intervention is to be done, and also how the logical setup of the network roughly is going to be. The latter is important to get a better understanding of how to physically place network equipment. Things that we need to buy for the next milestones should be known by the end of december, so that we can order it.

3.5.1.2 Physical

During the start of january, physical intervention can be started. At this point the logical structure of the network should be ready, and hence also a physical structure. Switches and cables is to be moved during this timeframe. Old hardware (both switches and cables) that isn't needed, should also be cleaned up.

3.5.1.3 Logical

Once the physical structure is done, we can configure the basic configuration of the switches on a logical level. This includes VLANs, routing, hostnames, IP-addresses, username/passwords, etc. During this period, central network-services should also be configured and set up – mainly DHCP, DNS and gateway servers. Since the LHC is to be commissioned by middle of march, the production-part of the network should be fully operational by end of february/start of march.

3.5.1.4 Monitoring

Once the network is more or less fully operational, we can start to implement monitoring tools. This is to get a better overview over the performance of the network, and to discover (and correct) flaws, potential bottlenecks, and errors.

3.5.1.5 Documentation

When monitoring is in place, and all other aspects of the network is done, the documentation-process can start. Documentation should be written "along-the-road", but should be combined into a complete documentation. A lot has probably also changed since it was written down, so it needs to be updated.

3.6 Risks

When doing such a large intervention as this, there are lots of things that can go wrong. Since we're basically pulling the plug on the network, and then building it from scratch, it can go wrong on both the logical and physical layer of the network, and you can also stumble upon weird software and/or hardware bugs. There are also other factors that can be an issue. I'll try to list most of them, and also give an estimation if they are risks that needs to be taken seriously or not.

Risk	**Probability vs. severity**	**Consequence**	**Comment**
Don't finish in time	10% / High	Can't join technical & physics-runs. Others might have to wait for us to be finished.	This should not happen. Have fallback-solution ready.
Cables break, physical connectivity-issues	40% / Medium	Machines and/or switches loses connectivity.	We have enough cables and patch-panels, so this is not an issue.
Switches/NICs fails	5% / High	Servers cannot communicate with the rest of the network.	We have spare switches and NICs, so this is not an issue.
Unforeseen requirements	30% / Medium	Can't proceed. Progress stops.	Hard to foresee everything in such a complicated setup as this.
Software, incompatibilities, bugs	10% / Medium	Can cause unwanted things to happen. Things might not work as expected, and can cause things to take long time to figure out.	Not very much likely to happen, and not much we can prepare for.
Sub-optimal configuration	30% / Medium	Things doesn't work as expected. Bad performance.	Issues can take long time to figure out if one is not 100% familiar with how the given feature is working.

Human error	80% / High	One can forget to do critical things. One can misconfigure things, which can make things take longer time. One can do things that has no effect at that point, but might yield unwanted results at a later time.	Cannot be avoided, as something is bound to be configured wrong in such a complicated setup. It shouldn't pose any risk towards not being able to finish the project, but can give bumps along the road.

Table 3.1: Risks - A list of the risks for the project.

3.7 Main goals

We have a few main goals, some of which the company has put forth. However, we've added some technical aspects to them, and this has also led to creation of additional goals.

3.7.1 Physical movement

The first thing we want to do, is to get rid of the centralized switch-placement. Instead of having all the switches in one place, we want to distribute them into the racks, and connect the servers directly to the switches (thus removing a source of error – the extra patching). This distribution also reduces the amount of cables needed at the core-switches.

3.7.2 Redundancy & availability

In addition to spread out the switches, we also want to increase the number of uplinks between all of the switches, and even more between the core-switches. This is both to ensure high bandwidth, but also to make sure we have enough redundancy; we should be able to lose any given cable, without loosing connectivity. Redundant paths is also something we want; we should be able to lose any given core-switch, without having noticeable downtime. We're also going to install two new gateways, which will be fully redundant (which isn't the case with the current gateways – they require manual intervention to be able to "failover").

3.7.3 IP-scheme

The current IP-scheme was poorly designed. It had a logic system that relied on bit-flipping. A complete re-do of the IP-scheme is wanted. Divide it up into subnets of suitable sizes (but still allow for future growth), and have a subnet for each kind of server/service.

3.7.4 Security

Security is also a concern – the cluster has previously, on several occasions, been rooted. This is partly caused by bad network-security (or rather, the lack of it), and partly by outdated, and poorly configured, Linux-installations. This is something we're going to fix. Updating the Linux-installations, using more secure gateways (OpenBSD), and generally limit the connections to the outside world.

3.7.5 Separation

Since the cluster consists of different "sections"; we have a production-cluster, development-cluster, infrastructure-machines, head-nodes, etc. While the cluster is in a running state, the production-cluster should be as isolated from the rest of the cluster as possible. This is to ensure that something, or someone, don't break things in the production-cluster while we're running – which is known to happen with the current setup.

3.7.6 Avoid loops

Loops has occurred in the network several times. We want to configure STP properly on all the switches, and ensure that it's working as intended.

3.7.7 Network-services

The current setup relies heavily on LDAP – users, DHCP, DNS, basically everything, is stored in the LDAP-database. This is fine, however, the learning-curve for LDAP is rather steep, so we're moving away from this. We're going to use move towards flat-files, as this simplifies things a lot, and we don't have to rely on yet another service that can fail. One less thing to maintain as well. We also want to set up proper DNS- and DHCP-services, together with the already mentioned redundant gateways.

4

Design

During this chapter, I'm going to describe the design of the network – in other words, this is the plan that the implementation-phase is going to be based upon. We can divide it into three parts; logical, physical and network-services.

4.1 Physical

There is some physical work to be done. To make things easier during implementation of new logical layout, we're going to do the physical changes first.

4.1.1 Switches

The switches are currently located in one place. We want to move the switches out to the different racks, so that 2 racks share 1 switch. This way we only need 4 uplink-cables to the core-switches. We're also going to have a lot of spacing in-between the core-switches, so that cable-management and troubleshooting becomes easy. We're going to use all 22 of the 3500yl-switches, and one of the 3400cl-48 switches, as our edge- and core-switches. The Netgear-switches is going to be used for management (IPMI, CHARM, BMC), just as in the old setup. The 3400cl-24 is still being used as our GPN-switch.

4.1.2 Cables

There are lots of cables. This is due to the fact that *all* switches are placed at the same place. To fix this, we are going to move the switches out in the different racks.

2 racks is going to share 1 switch. That 1 switch is going to have 4 uplinks to the core switches (3, in a trunk, to one, and the last one in another - this is to have full redundancy should a core-switch go down). This change is going to reduce the number of needed cables by a lot, going from around 2100 patch-cables (21 switches $*$ 48 ports $*$ 1.5 (since some patch-cables was connected through some etherlock-stuff)), to around 120 (18 edge-switches $*$ 4 uplinks + 4 core-switches $*$ 8 uplinks + 8 uplinks from infrastructure-switch). That's over 94% reduction of cables, which is going to ease up cable-troubleshooting by far.

Each core-switch is going to be interconnected with 8 uplinks, both to avoid bottlenecks, and to get redundancy, should cables fail. Each core-switch is going to be connected to both the two other cores. The core management-switch is being connected to the three cores with 2 uplinks, and the infrastructure-switch is getting 7 uplinks (in one trunk) to one core-switch, and 1 uplink to another (for redundancy).

To make sure I could be efficient when doing the recabling, I made a patch-list, with detailed description on where all the cables would go. The patch-list can be seen in Table 4.1. The "Trunk"-column is showing what trunk-number to use on that specific trunk. This is relevant when configuring the trunk on the switches.

From switch	Port(s)	Patchpanel	To switch	Port(s)	Trunk
sw-core0	1-8	-	sw-core2	1-8	trk1
sw-core0	9-16	-	sw-core1	1-8	trk2
sw-core0	17-18	-	sw-mgmt	1-2	trk10
sw-core0	19-21	CR2-Z02, 1-3	sw-cr2-z02	1-3	trk20
sw-core0	22-24	CR2-Z04, 1-3	sw-cr2-z04	1-3	trk21
sw-core0	25-27	CR2-Z06, 1-3	sw-cr2-z06	1-3	trk22
sw-core0	28-30	CR2-Z08, 1-3	sw-cr2-z08	1-3	trk23
sw-core0	31-33	CR2-Z10, 1-3	sw-cr2-z10	1-3	trk24
sw-core0	34-36	CR2-Z12, 1-3	sw-cr2-z12	1-3	trk25
sw-core0	37-39	CR3-X12, 1-3	sw-cr3-x12	1-3	trk26
sw-core0	40	CR2-Y05, 1	desk, dumb switch	-	-
sw-core0	42	-	sw-infra	8	-
sw-core0	43	CR2-Y06, 4	sw-cr2-y06	4	-
sw-core0	44	CR2-Y08, 4	sw-cr2-y08	4	-
sw-core0	45	CR2-Y10, 4	sw-cr2-y10	4	-
sw-core0	46	CR2-Y12, 4	sw-cr2-y12	4	-
sw-core0	47	CR2-Y14, 5	sw-cr2-y14	4	-

sw-core0	48	CR2-Y16, 4	sw-cr2-y16	4	-

sw-core1	1-8	-	sw-core0	9-16	trk2
sw-core1	9-16	-	sw-core2	9-16	trk3
sw-core1	17-18	-	sw-mgmt	3-4	trk11
sw-core1	19-21	CR2-X02, 1-3	sw-cr2-x02	1-3	trk27
sw-core1	22-24	CR2-X04, 1-3	sw-cr2-x04	1-3	trk28
sw-core1	25-27	CR2-X06, 1-3	sw-cr2-x06	1-3	trk29
sw-core1	28-30	CR2-X08, 1-3	sw-cr2-x08	1-3	trk30
sw-core1	31-33	CR2-X10, 1-3	sw-cr2-x10	1-3	trk31
sw-core1	34	CR2-Z02, 4	sw-cr2-z02	4	-
sw-core1	35	CR2-Z04, 4	sw-cr2-z04	4	-
sw-core1	36	CR2-Z06, 4	sw-cr2-z06	4	-
sw-core1	37	CR2-Z08, 4	sw-cr2-z08	4	-
sw-core1	38	CR2-Z10, 4	sw-cr2-z10	4	-
sw-core1	39	CR2-Z12, 4	sw-cr2-z12	4	-
sw-core1	40	CR3-X12, 4	sw-cr3-x12	4	-
sw-core1	41-47	-	sw-infra	1-7	trk4

sw-core2	1-8	-	sw-core0	1-8	trk1
sw-core2	9-16	-	sw-core1	9-16	trk3
sw-core2	17-18	-	sw-mgmt	5-6	trk12
sw-core2	19-21	CR2-Y06, 1-3	sw-cr2-y06	1-3	trk32
sw-core2	22-24	CR2-Y08, 1-3	sw-cr2-y08	1-3	trk33
sw-core2	25-27	CR2-Y10, 1-3	sw-cr2-y10	1-3	trk34
sw-core2	28-30	CR2-Y12, 1-3	sw-cr2-y12	1-3	trk35
sw-core2	31-33	CR2-Y14, 2-4	sw-cr2-y14	1-3	trk36
sw-core2	34-36	CR2-Y16, 1-3	sw-cr2-y16	1-3	trk37
sw-core2	37-38	-	gw1, em1 + em3	-	trk15
sw-core2	39	-	sw-gpn	21	-
sw-core2	44	CR2-X02, 4	sw-cr2-x02	4	-
sw-core2	45	CR2-X04, 4	sw-cr2-x04	4	-
sw-core2	46	CR2-X06, 4	sw-cr2-x06	4	-
sw-core2	47	CR2-X08, 4	sw-cr2-x08	4	-
sw-core2	48	CR2-X10, 4	sw-cr2-x10	4	-

sw-mgmt	1-2	-	sw-core0	17-18	trk10
sw-mgmt	3-4	-	sw-core1	17-18	trk11

sw-mgmt	5-6	-	sw-core2	17-18	trk12
sw-mgmt	7	CR2-X02, 24	sw-mgmt-cr2-x02	25	-
sw-mgmt	8	CR2-X03, 24	sw-mgmt-cr2-x03	25	-
sw-mgmt	9	CR2-X04, 24	sw-mgmt-cr2-x04	25	-
sw-mgmt	10	CR2-X05, 24	sw-mgmt-cr2-x05	25	-
sw-mgmt	11	CR2-X06, 24	sw-mgmt-cr2-x06	25	-
sw-mgmt	12	CR2-X07, 24	sw-mgmt-cr2-x07	25	-
sw-mgmt	13	CR2-X08, 24	sw-mgmt-cr2-x08	25	-
sw-mgmt	14	CR2-X09, 24	sw-mgmt-cr2-x09	25	-
sw-mgmt	15	CR2-X10, 24	sw-mgmt-cr2-x10	25	-
sw-mgmt	16	CR2-Y01, 24	sw-mgmt-cr2-y01	25	-
sw-mgmt	17	CR2-Y02, 24	sw-mgmt-cr2-y02	25	-
sw-mgmt	18	CR2-Y05, 24	sw-mgmt-cr2-y05	25	-
sw-mgmt	19	CR2-Y06, 24	sw-mgmt-cr2-y06	25	-
sw-mgmt	20	CR2-Y07, 24	sw-mgmt-cr2-y07	25	-
sw-mgmt	21	CR2-Y08, 24	sw-mgmt-cr2-y08	25	-
sw-mgmt	22	CR2-Y09, 24	sw-mgmt-cr2-y09	25	-
sw-mgmt	23	CR2-Y10, 24	sw-mgmt-cr2-y10	25	-
sw-mgmt	24	CR2-Y11, 24	sw-mgmt-cr2-y11	25	-
sw-mgmt	25	CR2-Y12, 24	sw-mgmt-cr2-y12	25	-
sw-mgmt	26	CR2-Y13, 24	sw-mgmt-cr2-y13	25	-
sw-mgmt	27	CR2-Y14, 24	sw-mgmt-cr2-y14	25	-
sw-mgmt	28	CR2-Y15, 24	sw-mgmt-cr2-y15	25	-
sw-mgmt	29	CR2-Y16, 24	sw-mgmt-cr2-y16	25	-
sw-mgmt	30	CR2-Y17, 24	sw-mgmt-cr2-y17	25	-
sw-mgmt	31	CR2-Z01, 24	sw-mgmt-cr2-z01	25	-
sw-mgmt	32	CR2-Z02, 24	sw-mgmt-cr2-z02	25	-
sw-mgmt	33	CR2-Z03, 24	sw-mgmt-cr2-z03	25	-
sw-mgmt	34	CR2-Z04, 24	sw-mgmt-cr2-z04	25	-
sw-mgmt	35	CR2-Z05, 24	sw-mgmt-cr2-z05	25	-
sw-mgmt	36	CR2-Z06, 24	sw-mgmt-cr2-z06	25	-
sw-mgmt	37	CR2-Z07, 24	sw-mgmt-cr2-z07	25	-
sw-mgmt	38	CR2-Z08, 24	sw-mgmt-cr2-z08	25	-
sw-mgmt	39	CR2-Z09, 24	sw-mgmt-cr2-z09	25	-
sw-mgmt	40	CR2-Z10, 24	sw-mgmt-cr2-z10	25	-
sw-mgmt	41	CR2-Z11, 24	sw-mgmt-cr2-z11	25	-
sw-mgmt	42	CR2-Z12, 24	sw-mgmt-cr2-z12	25	-
sw-mgmt	43	CR2-Z13, 24	sw-mgmt-cr2-z13	25	-
sw-mgmt	44	CR3-X11, 24	sw-mgmt-cr3-x11	25	-
sw-mgmt	45	CR3-X12, 24	sw-mgmt-cr3-x12	25	-

alihlt-swgpn	1-2	-	gw0, em0 + em2	-	trk16
alihlt-swgpn	3-4	-	gw1, em0 + em2	-	trk17
alihlt-swgpn	5	CR3-X12, 23	webcam, CR3-X12	-	-
alihlt-swgpn	6	CR2-X03, 1	webcam, CR2-X03	-	-
alihlt-swgpn	7	CR2-Y01, 1	not ours, Y01	-	-
alihlt-swgpn	8	CR2-X09, 1	webcam, CR2-X09	-	-
alihlt-swgpn	9	CR3-X05, 24	webcam, CR3-X05	-	-
sw-infra	1-7	-	sw-core1	41-47	trk4
sw-infra	8	-	sw-core0	42	-
sw-infra	9-10	-	gw0, em1 + em3	-	trk14

Table 4.1: Patchlist - The patchlist, showing where all cables are connected.

Taking into consideration the patchlist, and the physical placement of the switches, we start to see how everything connected. A preliminary network-diagram can be seen on Figure 4.1.

4.1.3 Servers

We need to move some servers, since they are going to be assigned new tasks (for some of the servers, this is just temporarily, until the virtual machines is in place). A few are going to be moved to act as temporary gateways, DHCP and DNS. The two latter is going to be virtualized at some point, while the gateways are going to be moved to some new servers in the future.

4.2 Logical

By logical, I mean IP-addresses and VLANs primarily. Services such as DNS, DHCP and the like, has been assigned it's own section at the end of this chapter.

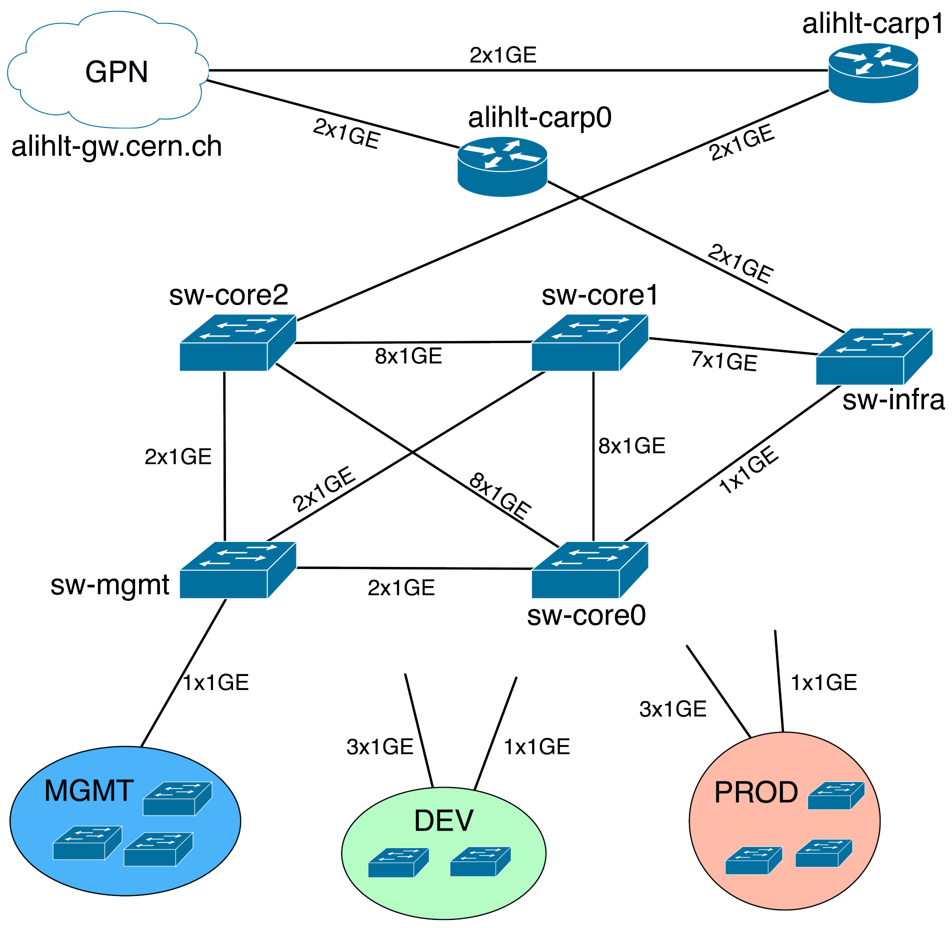

Figure 4.1: Network layout - Overview of the final network-layout.

4.2.1 IP-scheme

From CERN's IT-department, we've been assigned a class B subnet; 10.162.0.0/16. Even though all external connections to/from the cluster, is made via a gateway doing NAT (to public IP's), we chose to use this assigned range, just in case we ever need to interconnect our internal network with some of CERN's.

There is a need for three major subnets. One for InfiniBand to the PROD-cluster, one for ethernet (divided into smaller subnets), and one for management (IPMI, CHARM, BMC). Using this setup, we end up using 3/8 of available IP-addresses in the 10.162.0.0/16-network, which should give us plenty of room to grow in.

/16	/17	/18	/19	Description
10.162.0.0	10.162.0.0	10.162.0.0	10.162.0.0	Infiniband
			10.162.32.0	"Ethernet"
		10.162.64.0	10.162.64.0	Management
			10.162.96.0	Free
	10.162.128.0			Free

Table 4.2: Subnets - A table showing the different major subnets.

The "ethernet"-subnet is divided into smaller subnets.

/21	/22	/23	/24	Description
10.162.32.0				PROD-cluster
10.162.40.0				DEV-cluster
10.162.48.0	10.162.48.0			Infrastructure
	10.162.52.0			VMs
10.162.56.0	10.162.56.0	10.162.56.0	10.162.56.0	Gateway
			10.162.57.0	Login PROD-cluster
			10.162.58.0	Login DEV-cluster
			10.162.59.0	Admin
			10.162.60.0	Switch management
			10.162.61.0	Non-authorized hosts
			10.162.62.0	GPN management
			10.162.63.0	Free

Table 4.3: "Ethernet"-subnets - A table showing the different "ethernet"-subnets.

We have also pre-assigned IP-addresses for central services, such as gateways, DNS and DHCP. See Table 4.4.

Subnet	IP-address	Host
All subnets	10.162.x.1	sw-core1
	10.162.x.2	sw-core0
	10.162.x.3	sw-core2
	10.162.x.4	sw-mgmt
10.162.48.0/22	10.162.48.11	ns0
	10.162.48.12	ns1
	10.162.48.20	ecs0
	10.162.48.21	ecs1
	10.162.48.22	ecs
	10.162.48.30	dcs0
	10.162.48.31	dcs1
	10.162.48.32	dcs
	10.162.48.40	vobox0
	10.162.48.41	vobox1
	10.162.48.50	gui0
	10.162.48.51	gui1
	10.162.48.60	db0
	10.162.48.61	db1
	10.162.48.70	ms0
	10.162.48.71	ms1
	10.162.48.72	ms2
	10.162.48.80	mon0
	10.162.48.81	mon1
	10.162.50.2	vcm0
	10.162.50.10	vhost0
	10.162.50.11	vhost1
10.162.56.0/24	10.162.56.11	alihlt-carp0
	10.162.56.12	gw0
	10.162.56.13	gw1
10.162.59.0/24	10.162.59.11	desk
10.162.60.0/24	10.162.60.251	OSPF sw-mgmt
	10.162.60.252	OSPF sw-core2
	10.162.60.253	OSPF sw-core0
	10.162.60.254	OSPF sw-core1
10.162.62.0/24	10.162.62.11	sw-gpn

Table 4.4: Reserved IP-addresses - A table showing reserved IP-addresses.

4.2.2 VLANs

We need to put each of our subnets into it's own VLAN, so that we achieve separation on layer 2. This is also needed if we want to implement traffic-restrictions using access-lists at a later time.

VLAN ID	**Name**	**Subnet**	**Description**
1	DISABLED	-	VLAN is disabled, since this is the default VLAN.
5	NOT AUTHORIZED	10.162.61.0/24	For unused ports (and maybe future use together with 802.1X).
10	PROD	10.162.32.0/21	Production-cluster.
20	DEV	10.162.40.0/21	Development-cluster.
30	INFRA	10.162.48.0/22	Infrastructure. DHCP, DNS, storage, SQL, etc.
40	VM	10.162.52.0/22	Future subnet for VMs.
50	GW	10.162.56.0/24	For the main gateways providing internet.
60	LOGIN-PROD	10.162.57.0/24	Login-machines to the PROD-cluster.
70	LOGIN-DEV	10.162.58.0/24	Login-machines to the DEV-cluster.
80	SWITCH-MGMT	10.162.60.0/24	Management for the switches.
90	MGMT	10.162.64.0/19	Management for nodes (BMC, IPMI, CHARM).
100	ADMIN	10.162.59.0/24	Subnet for full access to entire network (when ACLs are implemented).
110	GPN	-	VLAN for the GPN-network on the GPN-switch.
120	GPN-MGMT	10.162.62.0/24	For managing the GPN-switch internally.

Table 4.5: VLANs - A table showing the different VLANs.

4.3 Network-services

The network-services is pretty much the same as before. The only exception is that we're moving away from LDAP. This is going to ease up on the maintenance, and the learning-curve isn't *that* steep.

4.3.1 DNS

We're going to run two physical servers, running Debian Squeeze with ISC DNS. They are going to be set up as master and slave, to ensure redundancy. At some point this service is going into two virtual machines, dedicated to running DNS (and nothing else).

4.3.2 DHCP

ISC DHCP is going to be set up on the same machines running ISC DNS. Only one subnet (NON AUTHORIZED HOSTS) is going to be set up with dynamic leases with redundancy. The rest are static leases, and doesn't need any redundancy (same static leases on both servers, first-come-first-serve rule applies). At some point, this service is going into two virtual machines, as with ISC DNS, but dedicated to running ISC DHCP (and nothing else - which implies 4 VMs in total for DNS and DHCP).

4.3.3 Gateway

This is going to be installed on two separate servers with a quad-NIC each, running OpenBSD. We're going to use PF and CARP to ensure NAT, firewall and redundancy/failover. We're going to trunk 2 links to achieve 2Gbps throughput.

5

Implementation

The implementation was done in several phases, according to our milestones as previously discussed.

5.1 Physical

First step was to re-cable and move switches. This took about 3 days, each with around 18 hours of work. First step was to power down the entire cluster, except critical infrastructure-machines (such as gateways, DHCP, DNS, fileservers, etc). I then wiped the config of two switches that was not in use, moved them out of the way, and moved all the previous mentioned infrastructure-machines over to this switch. This way I minimized the downtime of those nodes.

Next step was to pull all network-cables out from all the patchpanels and switches, and the power-cord from all the switches (except the two that the infrastructure was connected to, of course). I then moved one switch at the time, until all was moved. I then had 5 3500yl-switches left, together with some 3400cl's (that's going to be retired – the two temporary infrastructure-switches are part of these); 4 to be used as cores, and the last one to be used as infrastructure-switch (having extra bandwidth, since fileservers is to be connected to it).

Next up, was to move the core-switches a bit, so that they had more space between them (for easier access to cables), and to move the infrastructure-switch into place (almost the same place as the cores).

Now the patchlist (as seen on Figure 4.1) really started to come to use. First off was to interconnect all the core-switches and sw-infra (the infrastructure-switch). All the edge-switches was then to be connected to the core-switches. Then we had the gateways, the desk-machine, etc, that needed to be connected.

5.2 Switch configuration

After all the switches had been moved, and all the cables reconnected, it was time to reconfigure the switches.

5.2.1 Initial config

For the initial configuration, I made a template, with three variables; IP, host and trunk-number (as seen in the patchlist). This way I could generate config-files for all the switches, without having to manually change stuff – I just made a CSV-file (see Listing 5.1), containing a list of IP, host and trunk-number.

```
sw-cr2-x02#10.162.60.101#trk27
sw-cr2-x04#10.162.60.103#trk28
sw-cr2-x06#10.162.60.105#trk29
sw-cr2-x08#10.162.60.107#trk30
sw-cr2-x10#10.162.60.109#trk31
sw-cr2-y06#10.162.60.115#trk32
sw-cr2-y08#10.162.60.117#trk33
sw-cr2-y10#10.162.60.119#trk34
sw-cr2-y12#10.162.60.121#trk35
sw-cr2-y14#10.162.60.123#trk36
sw-cr2-y16#10.162.60.125#trk37
sw-cr2-z02#10.162.60.128#trk20
sw-cr2-z04#10.162.60.130#trk21
sw-cr2-z06#10.162.60.132#trk22
sw-cr2-z08#10.162.60.134#trk23
sw-cr2-z10#10.162.60.136#trk24
sw-cr2-z12#10.162.60.138#trk25
sw-cr3-x12#10.162.60.150#trk26
```

Listing 5.1: Switch CSV-file - One of the CSV-files used to generate switch-configs.

```
erase startup-config
config
no password manager
no password operator
max-vlans 20
wr mem
reload
conf
host HOSTLOL
spanning-tree
trunk 1-3 TRUNKLOL lacp
vlan 1 name DISABLED
no vlan 1 ip add
vlan 5 name NOT-AUTHORIZED
vlan 5 tagged TRUNKLOL,4
vlan 10 name PROD
vlan 10 tagged TRUNKLOL,4
vlan 20 name DEV
vlan 20 tagged TRUNKLOL,4
vlan 30 name INFRA
vlan 30 tagged TRUNKLOL,4
vlan 40 name VM
vlan 40 tagged TRUNKLOL,4
vlan 50 name GW
vlan 50 tagged TRUNKLOL,4
vlan 60 name LOGIN-PROD
vlan 60 tagged TRUNKLOL,4
vlan 70 name LOGIN-DEV
vlan 70 tagged TRUNKLOL,4
vlan 80 name SWITCH-MGMT
vlan 80 tagged TRUNKLOL,4
vlan 80 ip add IPLOL/24
vlan 90 name IPMI-CHARM
vlan 90 tagged TRUNKLOL,4
vlan 100 name ADMIN
vlan 100 tagged TRUNKLOL,4
vlan 200 name TEMP
vlan 200 tagged TRUNKLOL,4
vlan 200 untagged 5-48
no vlan 1 tagged TRUNKLOL,4-48
no vlan 1 untagged TRUNKLOL,4-48
```

Listing 5.2: Initial edge-switch config-template - The initial switch-config applied to all edge-switches.

With the config-template (see Listing 5.2) in place, together with the CSV-file, we can easily generate configs for all the switches, using a little bash-script (as seen in Listing 5.3). The script iterates through the CSV-file, replaces the temporary variable we set in the config-template, and saves each config as it's own file. We can then copy the content of a file, and paste it when configuring the appropriate switch.

```
#!/bin/bash

while read switch; do
  host=`echo "$switch"|cut -d'#' -f1`
  ip=`echo "$switch"|cut -d'#' -f2`
  trk=`echo "$switch"|cut -d'#' -f3`

  cat 3500yl-config-template.txt | sed -e "s/HOSTLOL/$host/" | sed -e "s/
      IPLOL/$ip/" | sed -e "s/TRUNKLOL/$trk/" > $host.txt
done < switch-ip-edge.txt
```

Listing 5.3: Bash-script to generate configs - Bash-script that generated all the necessary config-files.

The process to generate the initial configuration-files for the management-switches (se Listing 5.4 for the initial config, as this differs from the other, since it's a different switch-model) and core-switches, is just the same.

```
config
snmp-server name HOSTLOL
system config-pw-mode PC1-encrypted
vlan database
vlan 80 SWITCH-MGMT
vlan 90 MGMT
exit
spanning-tree 1w
spanning-tree priority 32768
system gateway 10.162.60.1
system ip-mode manual
system ip-addr IPLOL
system mask 255.255.255.0
management-vlan 80
```

```
interface ethernet 1/1
switchport access native 90
switchport access vlan untagged 90
exit

[ duplicate code removed, same config for ethernet 1/1 through 1/24 ]

interface ethernet 1/24
switchport access native 90
switchport access vlan untagged 90
exit


interface ethernet 1/25
switchport access native 1
switchport access vlan tagged 80
switchport access vlan tagged 90
exit


interface ethernet 1/26
switchport access native 1
switchport access vlan tagged 80
switchport access vlan tagged 90
exit

no system password
no system web

system save
```

Listing 5.4: Initial management-switch config-template - The initial switch-config applied to all management-switches.

5.2.2 Remove temporary switches

The switches used temporarily for the infrastructure, could now be removed. The infrastructure-machines was moved over to the new switches. As you can see in the configuration-files, all ports on all edge-switches has been put into VLAN 200, which is just a temporary VLAN without any routing. At this point, we actually powered on the entire cluster again. The nodes didn't notice the change – since it was placed on a VLAN, and the old network was just flat, this was all it needed to work. This way

I could configure more features on the switches, without causing too much downtime (and, if done at night, nobody would even notice).

5.2.3 Advanced switch config

With everything back up and working, we can configure more advanced services, mainly on the core-switches.

5.2.3.1 STP

Spanning Tree Protocol is used to avoid loops in the network. This can be fine-tuned, however, there are a few, basic settings that should be done. This is to ensure that the traffic flows in the right directions. On the edge-switches, the config shown in Listing 5.5.

```
spanning-tree <trunkname> path-cost 8000
spanning-tree 4 path-cost 20000
```

Listing 5.5: STP configuration on edge-switches - STP-config used on the edge-switches.

To see how STP was configured on the core-switches, see Listing A.2, A.3, A.4 and A.5.

5.2.3.2 Routing

To enable traffic to flow between the subnets/VLANs, we need to enable routing. This is as simple as turning it on, as seen on Listing 5.6. We could then route traffic between the subnets by using static routes, however, the smart thing to do, is to use a dynamic routing protocol, that keeps track of changes automatically.

```
ip routing
```

Listing 5.6: Turn on routing - Turning on routing on a switch is simple.

5.2.3.3 OSPF

As mentioned, we want to use a dynamic routing protocol to dynamically keep track of networks available. The setup is quite simple, as seen when configuring OSPF on sw-core1 in Listing 5.7.

```
ip route 0.0.0.0 0.0.0.0 10.162.56.11
ip route 10.162.0.0 255.255.224.0 reject
interface loopback 0 ip address 10.162.60.254
ip router-id 10.162.60.254
router ospf
area backbone
redistribute static
exit
vlan 5 ip ospf 10.162.61.1 area backbone
vlan 10 ip ospf 10.162.32.1 area backbone
vlan 20 ip ospf 10.162.40.1 area backbone
vlan 30 ip ospf 10.162.48.1 area backbone
vlan 40 ip ospf 10.162.52.1 area backbone
vlan 50 ip ospf 10.162.56.1 area backbone
vlan 60 ip ospf 10.162.57.1 area backbone
vlan 70 ip ospf 10.162.58.1 area backbone
vlan 80 ip ospf 10.162.60.1 area backbone
vlan 90 ip ospf 10.162.64.1 area backbone
vlan 100 ip ospf 10.162.59.1 area backbone
```

Listing 5.7: Configuring OSPF on sw-core1 - Configuring OSPF is quite simple.

The config for the other core-switches are similar, and can be seen in Listing A.3, A.4 and A.5.

5.2.3.4 VRRP

One of the biggest problems with using different subnets, is that you need a default gateway to be able to communicate with the other subnets. What happens if this gateway becomes unavailable, is that you can't reach any other subnets than the one you're in. One way to avoid this issue, is to configure VRRP, as this makes the gateway a virtual IP, that can be moved between the switches, so that, if a switch becomes unavailable, another switch takes over the role as gateway. Configuring this on sw-core1 can be seen in Listing 5.8.

```
router vrrp
vlan 5 vrrp vrid 5
  owner
    virtual-ip-address 10.162.61.1 255.255.255.0
    priority 255
    enable
    exit
  exit
vlan 10 vrrp vrid 10
    owner
    virtual-ip-address 10.162.32.1 255.255.248.0
    priority 255
    enable
    exit
  exit
vlan 20 vrrp vrid 20
    owner
    virtual-ip-address 10.162.40.1 255.255.248.0
    priority 255
    enable
    exit
  exit
vlan 30 vrrp vrid 30
    owner
    virtual-ip-address 10.162.48.1 255.255.252.0
    priority 255
    enable
    exit
  exit
vlan 40 vrrp vrid 40
    owner
    virtual-ip-address 10.162.52.1 255.255.252.0
    priority 255
    enable
    exit
  exit
vlan 50 vrrp vrid 50
    owner
    virtual-ip-address 10.162.56.1 255.255.255.0
    priority 255
    enable
    exit
  exit
vlan 60 vrrp vrid 60
```

```
    owner
    virtual-ip-address 10.162.57.1 255.255.255.0
    priority 255
    enable
    exit
  exit
vlan 70 vrrp vrid 70
    owner
    virtual-ip-address 10.162.58.1 255.255.255.0
    priority 255
    enable
    exit
  exit
vlan 80 vrrp vrid 80
    owner
    virtual-ip-address 10.162.60.1 255.255.255.0
    priority 255
    enable
    exit
  exit
vlan 90 vrrp vrid 90
    owner
    virtual-ip-address 10.162.64.1 255.255.224.0
    priority 255
    enable
    exit
  exit
vlan 100 vrrp vrid 100
    owner
    virtual-ip-address 10.162.59.1 255.255.255.0
    priority 255
    enable
    exit
  exit
```

Listing 5.8: Configuring VRRP on sw-core1 - Configuring VRRP is simple.

The config for the other core-switches are similar, and can be seen in Listing A.3, A.4 and A.5.

5.2.3.5 ip helper-address

To enable clients to reach DHCP-servers in other VLANs through DHCP-discover, `ip-helper-address <IP-address of DHCP-server>` is what you'll need to configure on the switches. This will forward the DHCP-discover to the IP-address listed, and the rest is normal procedure.

5.2.3.6 ip forward-protocol

We have a custom-made monitoring system in the cluster, that's called SysMES. It monitors all kinds of things (from network to filesystems), but it relies on broadcasts to work. Since not all SysMES-clients are in the same subnet (basically, all nodes is a SysMES-client), things won't work without extra configs on the switches. Here we need to specify `ip udp-bcast-forward` first, and then `ip forward-protocol <protocol> <destination IP-address> <port-number>` for each forward we want to have (and on each VLAN we want it working on). In our case, we have two SysMES-servers, which listens on messages broadcasted to destination-port 6666. The appropriate config is therefore as shown on Listing 5.9.

```
ip udp-bcast-forward
vlan 10 ip forward-protocol udp 10.162.48.30 6666
vlan 10 ip forward-protocol udp 10.162.48.31 6666
vlan 20 ip forward-protocol udp 10.162.48.30 6666
vlan 20 ip forward-protocol udp 10.162.48.31 6666
vlan 30 ip forward-protocol udp 10.162.48.30 6666
vlan 30 ip forward-protocol udp 10.162.48.31 6666
vlan 40 ip forward-protocol udp 10.162.48.30 6666
vlan 40 ip forward-protocol udp 10.162.48.31 6666
vlan 50 ip forward-protocol udp 10.162.48.30 6666
vlan 50 ip forward-protocol udp 10.162.48.31 6666
vlan 60 ip forward-protocol udp 10.162.48.30 6666
vlan 60 ip forward-protocol udp 10.162.48.31 6666
vlan 70 ip forward-protocol udp 10.162.48.30 6666
vlan 70 ip forward-protocol udp 10.162.48.31 6666
vlan 80 ip forward-protocol udp 10.162.48.30 6666
vlan 80 ip forward-protocol udp 10.162.48.31 6666
vlan 90 ip forward-protocol udp 10.162.48.30 6666
vlan 90 ip forward-protocol udp 10.162.48.31 6666
vlan 100 ip forward-protocol udp 10.162.48.30 6666
vlan 100 ip forward-protocol udp 10.162.48.31 6666
```

Listing 5.9: Configuring broadcast-forwards - Making the switch forward broadcasts to a specific IP-address.

5.2.4 Final switch-configs

The final configuration for sw-core1 can be seen in Listing A.2. The final configuration for sw-core0 can be seen in Listing A.3. The final configuration for sw-core2 can be seen in Listing A.4. The final configuration for sw-mgmt can be seen in Listing A.5.

5.3 DNS & DHCP

After the configuration of the switches was done, we could start on the configuration of DHCP and DNS.

5.3.1 Preparation

First of all, we need to extract the old information from the old LDAP-database (that we don't use anymore), so that we don't have to manually enter all the MAC-addresses again. Since the old LDAP-database had a lot of information, it was a somewhat tedious process to extract the correct information.

First, I extracted the hostname, IP-address and MAC-address, from the old LDAP-database. I excluded all entries that had an MAC-address of "00:00:00:00:00:00", due to the fact that it's not a valid MAC-address (for a node – it's used in some rare occasions for broadcasting, so in a way it's a valid MAC-address), and that all entries without any associated host had this value as it's MAC-address (which actually could be used to find free/available IP-addresses in the old setup – I guess they didn't think of that). I then put it all into a CSV-file. This was all done using the commands as seen in Listing 5.10.

```
cat ldap.ldif | grep -A18 "ou=dhcp" > ldap-dhcp.ldif

cat ldap-dhcp.ldif | grep -i "ou=dhcp" | cut -d',' -f1 | sed -e 's/dn\: cn
    \=//' | grep -viE "(dhcp|192.168.0.0|group1|pool1)" > hostnames.txt

while read hostname; do
  ip=`cat ldap-dhcp.ldif | grep -A18 "cn\=$hostname\," | grep "fixed\-
      address" | sed -e 's/dhcpStatements\:\ fixed\-address\ //'`
  mac=`cat ldap-dhcp.ldif | grep -A18 "cn\=$hostname\," | grep "
      dhcpHWAddress" | sed -e 's/dhcpHWAddress\:\:\ //' | sed -e 's/
      dhcpHWAddress\:\ ethernet\ //'`
  if [ ! -z "$ip" ]; then
    # $ip has a value

    echo "$hostname # $mac # $ip" >> complete-address-list.txt

    if [ "$mac" != "00:00:00:00:00:00" ]; then
      # only print entries that has a valid MAC-address

      echo "$hostname # $mac # $ip" >> only-valid-mac-addresses.txt
    fi
  fi
done < hostnames.txt
```

Listing 5.10: Extracting information from old LDAP-database - The commands used to extract IP, MAC and hostname from the old LDAP-database.

Then I had to process this CSV-file, and generate DNS and DHCP configuration-files with new hostnames and IP-addresses (the MAC is still the same). This was done with a lot of trying and failing. After a lot of ugly bash-scripting, I managed to produce usable results. You can see the final script (it could have been written *much* cleaner, though) in Listing A.1.

5.3.2 Final config

After having generated these configuration-files containing host-information, it was just a matter of finalizing it all, using pretty much just standard configurations.

5.4 Gateway

The gateways was configured temporarily (they are to be moved to virtual machines at a later point) on two way too powerful nodes (you don't need 16 cores and 24GB of memory to run DHCP and DNS, unless you have a godlike amount of nodes). I installed OpenBSD 4.8 on them, and configured them with PF (NAT and firewall) and CARP (for redundancy).

5.4.1 Initial network-config

Before I could start configuring CARP and PF, I needed to have basic network configuration in place. Since we're going to trunk two links to get 2Gbps throughput, we need to set up this first. We have a quad-NIC in each gateway, giving us a total of 6 available Gpbs network-interfaces. We're going to use two for the external network, two for the internal network, and one to sync PF between the two gateways. To achieve added redundancy, we're going to put one internal and one external network-interface in each trunk. The configuration shown in Listing 5.11 is for the master gateway – the configuration is just the same for the backup gateway, except different IP-addresses.

```
# Configure working config on running system
root@alihlt-carp0:~# ifconfig em0 up
root@alihlt-carp0:~# ifconfig em1 up
root@alihlt-carp0:~# ifconfig em2 up
root@alihlt-carp0:~# ifconfig em3 up
root@alihlt-carp0:~# ifconfig trunk0 trunkport em0
root@alihlt-carp0:~# ifconfig trunk0 trunkport em2
root@alihlt-carp0:~# ifconfig trunk0 trunkproto lacp 137.138.11.20 netmask
     255.255.0.0
root@alihlt-carp0:~# ifconfig trunk0 trunkport em1
root@alihlt-carp0:~# ifconfig trunk0 trunkport em3
root@alihlt-carp0:~# ifconfig trunk0 trunkproto lacp 10.162.56.12 netmask
    255.255.255.0

# To make it persistant through reboot
root@alihlt-carp0:~# for nic in em0 em1 em2 em3; do echo "up" > /etc/
    hostname.$nic; done
root@alihlt-carp0:~# echo "trunkproto lacp trunkport em0 trunkport em2
    137.138.11.20 netmask 255.255.0.0" > /etc/hostname.trunk0
root@alihlt-carp0:~# echo "trunkproto lacp trunkport em1 trunkport em3
    10.162.56.12 netmask 255.255.255.0" > /etc/hostname.trunk1

# We should also add a routing-entry to let the
# gateway know about the 10.162.0.0/16-network
root@alihlt-carp0:~# route add -net 10.162.0.0/16 10.162.56.1
root@alihlt-carp0:~# echo "!route add -net 10.162.0.0/16 10.162.56.1" >> /
    etc/hostname.trunk1
```

Listing 5.11: Basic network-config on gateways - Commands to implement basic network configuration when using trunks.

We also have to configure this on the switches, since we're trunking. This was already done before configuring the gateways.

5.4.2 CARP

Once normal network-connectivity is achieved, we can start configuring CARP. See Listing 5.12 for a complete configuration.

```
# Make alihlt-carp0 master, and after a failover, make it master again
root@alihlt-carp0:~# sysctl net.inet.carp.preempt=1
net.inet.carp.preempt: 0 -> 1

# Make it persistent through boot, add the following to
# /etc/sysctl.conf (uncomment it if it's already there)
# On the backup (alihlt-carp1), make sure it's _NOT_ there
root@alihlt-carp0:~# cat /etc/sysctl.conf|grep preempt
net.inet.carp.preempt=1 # 1=Enable carp(4) preemption

# Then, let's set up the CARP-interfaces
# (two, in our case - internal and external)
root@alihlt-carp0:~# ifconfig carp0 137.138.11.19/16 vhid 55 pass
    supersecret advbase 1 advskew 0
root@alihlt-carp0:~# ifconfig carp1 10.162.56.11/24 vhid 66 pass
    supersecret advbase 1 advskew 0
root@alihlt-carp0:~# ifconfig carp0
carp0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        lladdr 00:00:5e:00:01:37
        priority: 0
        carp: MASTER carpdev trunk0 vhid 55 advbase 1 advskew 0
        groups: carp
        status: master
        inet 137.138.11.19 netmask 0xffff0000 broadcast 137.138.255.255
        inet6 fe80::200:5eff:fe00:137%carp0 prefixlen 64 scopeid 0xb
root@alihlt-carp0:~# ifconfig carp1
carp1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        lladdr 00:00:5e:00:01:42
        priority: 0
        carp: MASTER carpdev trunk1 vhid 66 advbase 1 advskew 0
        groups: carp
        status: master
        inet 10.162.56.11 netmask 0xffffff00 broadcast 10.162.56.255
        inet6 fe80::200:5eff:fe00:142%carp1 prefixlen 64 scopeid 0xc
```

```
# Now, we need to make it persistent through reboots
root@alihlt-carp0:~# echo "inet 137.138.11.19 255.255.0.0 137.138.255.255
    vhid 55 pass <supersecret> advbase 1 advskew 0" > /etc/hostname.carp0
root@alihlt-carp0:~# echo "inet 10.162.56.11 255.255.255.0 10.162.56.255
    vhid 66 pass <supersecret> advbase 1 advskew 0" > /etc/hostname.carp1
```

Listing 5.12: CARP-configuration - Configuration of CARP in OpenBSD 4.8.

Now, we apply the exactly the same configuration on the other gateway. The only thing we change, is the value of "advskew"; the higher this number is, the lower priority it gets. The lower priority it has, the lower chance it has of becoming master. In other words; the CARP-device with the highest adskew-value, becomes master (it's based on a formula, so, you can also change "advbase", but this leads to longer time before a backup takes over in case the master goes down). I've used 100 as the adskew-value on the backup-gateway.

5.4.3 PF

Now, after CARP is done, we need to setup the firewall itself, PF – which will handle NAT, firewall-rules, port-forwards, and such things. Since we're using CARP, we also need to keep the state of all connections in sync between the gateways (so that if a gateway fails, the clients won't notice anything).

To do this, we've connected em4 on one gateway, to em4 on the other (directly, it does not go through any switches or anything like that). This is done because the pfsync protocol doesn't provide any cryptography or authentication mechanism – this, if you don't use a secure network, like a crossover cable, an attacker may use spoofed pfsync packets to alter the firewalls state tables and bypass filter rules. Se Listing 5.13 for the configuration needed.

```
# On main gateway
root@alihlt-carp0:~# ifconfig em4 172.30.30.1 netmask 255.255.255.0
root@alihlt-carp0:~# ifconfig pfsync0 syncdev em4
root@alihlt-carp0:~# ifconfig pfsync0 up
root@alihlt-carp0:~# ifconfig pfsync0
pfsync0: flags=41<UP,RUNNING> mtu 1500
```

```
        priority: 0
        pfsync: syncdev: em4 maxupd: 128 defer: off
        groups: carp pfsync
root@alihlt-carp0:~# echo "inet 172.30.30.1 255.255.255.0 172.30.30.255" >
    /etc/hostname.em4
root@alihlt-carp0:~# echo "up syncdev em4" > /etc/hostname.pfsync0

# On backup gateway
root@alihlt-carp1:~# ifconfig em4 172.30.30.2 netmask 255.255.255.0
root@alihlt-carp1:~# ifconfig pfsync0 syncdev em4
root@alihlt-carp1:~# ifconfig pfsync0 up
root@alihlt-carp1:~# ifconfig pfsync0
pfsync0: flags=41<UP,RUNNING> mtu 1500
        priority: 0
        pfsync: syncdev: em4 maxupd: 128 defer: off
        groups: carp pfsync
root@alihlt-carp1:~# echo "inet 172.30.30.2 255.255.255.0 172.30.30.255" >
    /etc/hostname.em4
root@alihlt-carp1:~# echo "up syncdev em4" > /etc/hostname.pfsync0
```

Listing 5.13: PF-setup - Setup of PF in OpenBSD 4.8.

Now, to the fun part. Configuring the actual PF-configuration file. Since we're using CARP, we should have the same PF-configuration on both gateways (if the master-gateway goes down, we want the backup-gateway to behave just the same). We can make the configuration on the master-gateway, and then just copy it directly over to the backup-gateway (without changing anything at all, actually). The current PF-config, which is stored in /etc/pf.conf, is shown in Listing 5.14.

```
#### Macros & variables
# Interfaces
pfsync_if="em4"
ext_if="trunk0"
int_if="trunk1"
ext_carp_if="carp0"
int_carp_if="carp1"
all_ext_if="{" $ext_if $ext_carp_if "}"
all_int_if="{" $int_if $int_carp_if "}"
all_if="{" $ext_if $ext_carp_if $int_if $int_carp_if "}"

# IPs
```

```
ext_dns_srv="{ 137.138.16.5, 137.138.17.5 }"
int_dns_srv="{ 10.162.48.11, 10.162.48.12, 10.162.50.61, 10.162.50.62 }"
ext_gw="137.138.1.1"
smtp_int="10.162.50.78"
hlt_prod="10.162.57.11"
hlt_dev="10.162.58.11"

# Allowed ICMP-types
icmp_types="{ echorep, echoreq, timex, paramprob, unreach code needfrag }"

# Blocked nets
table <blocked_nets> { 127.0.0.0/8, 192.168.0.0/16, 172.16.0.0/12,
    10.0.0.0/8, 169.254.0.0/16, 192.0.2.0/24, 0.0.0.0/8, 240.0.0.0/4 }

# Our networks
our_int_net="{ 10.162.0.0/16 }"
table <not_our_int_net> { !10.162.0.0/16 }

#### Options and NAT
# Packets that are blocked, will be dropped
set block-policy drop

# Log things if specified in filters
set loginterface $ext_carp_if

# Skip filtering on loopback-interface(s)
set skip on lo

# NAT all outgoing connections
match out log on $ext_if inet from $our_int_net to any nat-to (
    $ext_carp_if)

#### Redirection / port-forward
# SSH/NX to hlt-prod
pass in quick log on $ext_if proto tcp from any to any port 122 rdr-to
    $hlt_prod port 22

# SSH/NX to hlt-dev
pass in quick log on $ext_if proto tcp from any to any port 222 rdr-to
    $hlt_dev port 22

#### Filtering rules
# Default deny
block in log
```

```
# Spoofed address protection
block in quick log from urpf-failed

# Scrub incoming packets
match in all scrub (no-df)

# Enable pfsync
pass quick log on $pfsync_if proto pfsync keep state (no-sync)

# Enable CARP
pass quick log on { $ext_if , $int_if } proto carp keep state (no-sync)

# Block request from outside with blocked addresses
block in  quick log on $all_ext_if from <blocked_nets> to any

# Block request from inside to blocked addresses
block out quick log on $all_ext_if from any to <blocked_nets>

# Default pass outgoing
pass out log keep state

## Pass local net
# Pass everything from local net
pass in on $int_if from $our_int_net

## Pass from local net, before block
# Allow outgoing SMTP from this host
pass in quick log on $int_if inet proto { tcp, udp } from $smtp_int to any
     port smtp

## Block from local net
# Allow outgoing SMTP from this host
block in log on $int_if inet proto { tcp, udp } from any to any port smtp

# ICMP
pass in quick inet proto icmp all icmp-type $icmp_types

# SSH
pass in quick on $all_ext_if inet proto tcp from any to $all_ext_if port
    ssh
```

Listing 5.14: PF-configuration - Configuration of PF filters & rules.

There is still room for improvement in the PF-configuration (stricter rules/filters, etc), however, this gives us a working base-configuration, which can be improved later on.

One more thing we'd like to do, is to make sure PF keeps it's logs for more then a few hours. We should change a few entries in /etc/newsyslog.conf, as shown in Listing 5.15.

```
# In /etc/newsyslog.conf, change the file
# /var/log/pflog-entry to something like

/var/log/pflog 600 100 102400 * ZB "pkill -HUP -u root -U root -t - -x
    pflogd"
```

Listing 5.15: PF-logging - Change how long PF should keep its logs.

5.5 The switchover

After all configuration was done, I decided to move things over from the old network, to the new network, in three phases. The only thing I needed to do, was to change the VLAN of the access-ports accordingly. First the management-network, as this could be done without causing downtime in the cluster. This went rather smoothly, except that some of the nodes had BMC-cards that only sent DHCP-discover through a single interface (which, in turn, forced me to change the main interface to the internal network on these nodes). After testing that it was working as intended, I could move on to the next step, which was the production-cluster. This also went rather smoothly. The development-cluster was not moved, since we wanted to reinstall these nodes. This could not be done before other services in the network was ready (such as PXE-booting and SVN-repositories containing the node-configuration files).

6

Testing

During, and especially after, implementation, we want to test that everything works according to the plan.

6.1 Network-connectivity

Several network-connectivity tests was performed regularly during the switchover-phases, to make sure we had connectivity to all nodes. This was mainly done using a hostname-list, and iterate through it, pinging each host. Example of a typical command, can be seen in Listing 6.1.

```
root@portal-ecs1:~# for host in $(cat /etc/dsh/group/prodcluster); do if !
     ping -c1 -w1 $host|grep -qi "bytes from"; then echo "the host $host
    is unreachable"; fi; done
the host cn010 is unreachable
the host fephmpid3 is unreachable
the host fepsdd1 is unreachable
the host fepsdd5 is unreachable
the host feptofa08 is unreachable
```

Listing 6.1: Verifying working-connectivity - Making sure the network is working.

In most cases, as you could see in Listing 6.1, there was nodes not responding. Usually, these where hosts that we already knew was offline (due to maintenance).

6.2 Network-services

There are several network-services that we rely heavily on. There are three main network-services that we have discussed.

6.2.1 DHCP

After configuring 'ip helper' on the core-switches, we want to make sure that the DHCP-servers gets the requests. This can be done by checking the log-files, as shown in Listing 6.2.

```
root@ns0:/etc/bind# cat /var/log/syslog|grep "REQUEST"|grep "via 10"|grep
    -v "free"|tail -3
Jun  9 21:01:40 ns0 dhcpd: DHCPREQUEST for 10.162.82.10 from 00:40:dc:fa
    :00:af via 10.162.64.4
Jun  9 21:03:44 ns0 dhcpd: DHCPREQUEST for 10.162.64.128 from 00:e0:81:4d
    :2c:49 via 10.162.64.2
Jun  9 21:16:46 ns0 dhcpd: DHCPREQUEST for 10.162.72.34 from 00:40:dc:fa
    :00:5f via 10.162.64.3
```

Listing 6.2: Verifying working DHCP - Making sure the DHCP-server gets requests from other VLANs besides its own.

6.2.2 DNS

DNS was put to test each time we ran the previous ping-tests (since they use host-names). We also verified it manually several times, both forward and reverse-entries, as shown in Listing 6.3.

```
root@portal-ecs1:~# dig feptofa02.internal a +short
10.162.32.139
root@portal-ecs1:~# dig -x 10.162.32.139 +short
feptofa02.internal.
root@portal-ecs1:~# dig alihlt-gw.internal a +short
10.162.56.11
root@portal-ecs1:~# dig -x 10.162.56.11 +short
alihlt-gw.internal.
```

Listing 6.3: Verifying working DNS - Making sure the DNS-servers is working.

6.2.3 Gateways

Testing of the gateways was not done specifically, as all communication to and from the cluster is done via the gateway. There was no discovery of services not working, so there was no immediate need to do any more testing of this.

6.3 Redundancy

Testing of redundancy could only be done after all aspects of the network was implemented (STP, VRRP, CARP, etc).

6.3.1 Edge-switches

Edge-switches had one, two or three cables pulled out in different variations. There are two links (3x1Gpbs and 1x1Gpbs), and pulling either of these, or just single cables in the trunk, was tested. Downtime was the little time that STP spends converging (usually non-noticeable for the clients).

6.3.2 Core-swithces & gateways

I did some failover-tests, which basically included rebooting the master-gateway, and the master core-switch (sw-core1), in addition to the infra-switch (all at the same time). The total downtime was 3-4 seconds for the failover, and around 1 second for the pre-empt (failback). The failover/pre-empt will not cause any issues for users (except that they might notice a slight ”glitch” – however, it will not disconnect sessions, etc).

The 3-4 seconds for the failover is basically a limitation in how VRRP works; the master-router has an internal advertisement-timer, where the lowest value is 1 second. A backup-router will only take over if it doesn't receive an advertisement for 3x the advertisement-timer – in other words; a backup-router won't take over for a failed master-router before it has gone 3 seconds.

6.3.3 Network-services

Shutting down either of the servers running DNS and DHCP did not affect the service. However, since both nameservers are on the same switch, we'll loose DNS-lookups if the 'sw-infra'-switch fails. Since we're moving the nameservers to virtual machines, this issue will be resolved at that point. The solution will simply be to put one of the virtualization-servers, hosting one of the nameservers, on a different switch than the virtualization-server that hosts the other nameserver. This actually applies for the DHCP-servers as well – however, clients won't loose their IP once they've first have it.

7 Conclusion

We have built a scalable, stable and fully redundant network, with a working IP-scheme, and network-services that is easy to manage.

7.1 Fulfilled requirements

All the requirements that the company had, and those we specified extra on this project, was fulfilled. One can perhaps argue that the separation-requirement hasn't been fulfilled, which is somewhat true (it's separated at layer 2, but not on layer 3), however, this can easily be fixed using some ACLs, which we'll talk about below.

7.2 Limitations

There might be limitations in some of the services and/or protocols involved, however, none has come to our attention. Other than that, there are no known limitations.

7.3 Improvements

There is still lots of improvements that can be done. We can fine-tune many of the services we've implemented. I'll list those that has come to our mind here.

- Implement separation at layer 3 between production- and development-cluster.
- Implement ACLs to improve security in general. Restrict access to different parts of the network from certain nodes.

- Buy some more switches, so that we have unison model-numbers (that is, buy 3500yl's to replace 3400cl's). This makes it easier to replace broken switches, and do configuration-changes, since the syntax will be the same (which is not the case now, when we have both 3500yl's and 3400cl's).
- Buy some Supermicro Atom-nodes to replace the current gateways, as they don't need the processing-power they currently have.
- Buy some Supermicro Atom-nodes to act as gateways between our network, and the DCS/ECS-networks.
- Script that parses a CSV-file, containing hostname, IP-address and MAC-address, into configuration-files for DHCP and DNS, so that changes can be done easily. This way you'd only need to edit one file to change it in both DNS and DHCP.
- More strict PF-rules.
- Fine-tune STP.
- Move network-services over to virtualized machines, with redundancy regarding switches (two VM's running on two different physical machines, connected to two different switches).
- Make some more logic to the PTR-record on the default gateways. Minor detail, but somewhat important since we're using VRRP (default gateway for the INFRA-subnet is 10.162.48.1, which has a PTR-record to sw-core1-vlan30.internal. If sw-core1 goes down, sw-core0 takes over, and then the PTR-record doesn't make sense).

7.4 Progress according to plan

All milestones was completed according to the plan.

References

[1] **A Large Ion Collider Experiment - Wikipedia** [online, cited Wednesday, 08. June, 2011]. x

[2] **Autonomous system (Internet) - Wikipedia** [online, cited Wednesday, 08. June, 2011]. x

[3] **Baseboard management controller - Wikipedia** [online, cited Wednesday, 08. June, 2011]. x

[4] **Broadcast radiation - Wikipedia** [online, cited Wednesday, 08. June, 2011]. x

[5] **PF - Firewall Redundancy with CARP and pfsync** [online, cited Wednesday, 08. June, 2011]. x, xii

[6] **Common Address Redundancy Protocol - Wikipedia** [online, cited Wednesday, 08. June, 2011]. x

[7] **CERN - Wikipedia** [online, cited Wednesday, 08. June, 2011]. x

[8] **The ALICE High Level Trigger** [online, cited Wednesday, 08. June, 2011]. x, xi

[9] **Fernadministration CHARM** [online, cited Wednesday, 08. June, 2011]. x

[10] **Dynamic Host Configuration Protocol - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xi

[11] **Domain Name System - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xi

[12] **EtherLock II Product Family - CMS Technologies** [online, cited Wednesday, 08. June, 2011]. xi

[13] **Gateway (telecommunications) - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xi

[14] **InfiniBand - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xi

[15] **IP address - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xi

[16] **Intelligent Platform Management Interface - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xi

[17] **Internet Systems Consortium - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xi

[18] **MAC address - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xi

[19] **Network address translation - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xii

[20] **Switching loop - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xii

[21] **Node - Definition for Node** [online, cited Wednesday, 08. June, 2011]. xii

[22] **OSI model - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xii

[23] **Open Shortest Path First - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xii

[24] **PF (firewall) - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xii

[25] **Rack unit - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xii

[26] **Rootkit - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xiii

[27] **Spanning Tree Protocol - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xiii

[28] **Subnetwork - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xiii

[29] **Throughput - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xiii

[30] **Link aggregation - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xiii

[31] **Time to live - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xiii

[32] **Virtual LAN - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xiii

[33] **Virtual Router Redundancy Protocol - Wikipedia** [online, cited Wednesday, 08. June, 2011]. xiv

[34] **ProCurve Switch 3400cl-24G (J4905A)** [online, cited Wednesday, 08. June, 2011]. 7

[35] **ProCurve 10-GIG Media Flex Module (J8435A)** [online, cited Wednesday, 08. June, 2011]. 7

[36] **HP 10GbE X2-SC LR Optic (J8437A)** [online, cited Wednesday, 08. June, 2011]. 7

[37] **ProCurve Switch 3400cl-48G (J4906A)** [online, cited Wednesday, 08. June, 2011]. 8

[38] **ProCurve Switch 3500yl-48G-PWR (J8693A)** [online, cited Wednesday, 08. June, 2011]. 8

[39] **Netgear FSM726v2** [online, cited Wednesday, 08. June, 2011]. 8

[40] **HP E3500 Switch Series** [online, cited Wednesday, 08. June, 2011]. 8

[41] **HP ProCurve** [online, cited Friday, 10. June, 2011].

[42] **HP.com** [online, cited Friday, 10. June, 2011].

[43] **New HLT Wiki** [online, cited Friday, 10. June, 2011].

[44] **Old HLT Wiki** [online, cited Friday, 10. June, 2011].

[45] **cern.jocke.no** [online, cited Wednesday, 08. June, 2011].

[46] **ALICE - A Large Ion Collider Experiment** [online, cited Wednesday, 08. June, 2011].

[47] **PF - The OpenBSD Packet Filter** [online, cited Thursday, 09. June, 2011].

[48] **Classful network - Wikipedia** [online, cited Thursday, 02. June, 2011].

[49] **IPv4 - Wikipedia** [online, cited Thursday, 02. June, 2011].

Appendix A

Scripts & config

```
#!/bin/bash

# Variables
ip_start="10.162"
ib_mac="99:99:99:99:99:99"

# Switches
rename_cn=1
rename_fep=0
print_stuff=0
generate_config=1

# Remove unwanted files
rm dhcpd-10.162.* 2> /dev/null
rm db.master_* 2> /dev/null

die (){
        echo >&2 "$@"
        exit 1
}

increment (){
  stripped_hostname=`echo $new_hostname|perl -wple 's/(\-).?(charm|bmc|ib)
      $//ig'`

  if [ -z "$last_hostname" ]; then
          # This is the first hostname we're processing
          last_hostname="$stripped_hostname"
    fi
    if ! echo "$stripped_hostname"|grep -qiE "^$last_hostname$"; then
                increment_ip
    fi

    last_hostname=$stripped_hostname
}


increment_ip (){
  if [ $fourth_octet -eq 255 ]; then
                # Increase third octet, reset fourth
                let "third_octet += 1"
                fourth_octet=1
        else
    # Just increase fourth
                let "fourth_octet += 1"
        fi
}

generate_bind_special (){
  # Called when we want to iterate through a file with complete IPs/MACs
  # $1 = filename
  # $2 = network_name

  bind_conf_fwd_filename="db.master_internal_$2"
  bind_conf_ptr_filename="db.master_10-162-0-0_$2"

  while read hostinfo; do
    hostname=`echo "$hostinfo"|cut -d'#' -f1`
    ip=`echo "$hostinfo"|cut -d'#' -f2`

    generate_bind $hostname $ip
  done < $1
}

generate_config (){
  # Called when we want to generate configuration

  # $1 = hostname
  # $2 = mac
  # $3 = ip
  # $4 = infinihost

  generate_dhcp $1 $2 $3 $4
  generate_bind $1 $3
```

```
}

generate_dhcp (){
  # Generate the DHCP-config

  if [ -z $4 ]; then
    # We don't want infinihosts in the DHCP-config

    octet_3=`echo "$3"|cut -d'.' -f3`

    if [ $octet_3 -ge 64 ] && [ $octet_3 -lt 96 ]; then
      # It's CHARM or BMC
      echo "host $1 { hardware ethernet $2; fixed-address $3; }" >>
          $dhcp_conf_filename_mgmt
    else
      # It's a host
      echo "host $1 { hardware ethernet $2; fixed-address $3; }" >>
          $dhcp_conf_filename
    fi
  fi
}

generate_bind (){
  # Generate the BIND-config
  octet_3=`echo "$2"|cut -d'.' -f3`
  octet_4=`echo "$2"|cut -d'.' -f4`

  # Generate forward
  printf "%-25s%-5s%-5s%-15s\n" "$1" "IN" "A" "$2" >>
      $bind_conf_fwd_filename

  # Generate reverse
  printf "%-10s%-5s%-10s%-20s\n" "$octet_4.$octet_3" "IN" "PTR" "$1.
      $domain_name." >> $bind_conf_ptr_filename

  # Generate alias
  if echo "$1"|grep -qiE "(bmc|charm)"; then
    stripped=`echo "$1"|perl -wple 's/\-.?(bmc|charm)//gi'`
    printf "%-25s%-5s%-10s%-20s\n" "$stripped-mgmt" "IN" "CNAME" "$1.
        $domain_name." >> $bind_conf_alias_filename
  fi
}

run_code (){
```

```
# Variables
static_third_octet=$2
third_octet=$static_third_octet
fourth_octet=$3
ib_offset="$4"
mgmt_offset="$5"
cn_counter=0
cngpu_counter=0
fep_counter=0
fephost_without_number="LOLKEK"
dhcp_conf_filename="dhcpd-10.162.$static_third_octet.0.conf"
dhcp_conf_filename_mgmt="dhcpd-10.162.64.0.conf"
bind_conf_fwd_filename="db.master_internal_$6"
bind_conf_ptr_filename="db.master_10-162-0-0_$6"
bind_conf_alias_filename="db.master_internal_alias"
domain_name="internal"
last_hostname=""

while read hostinfo; do
  old_hostname=`echo $hostinfo | cut -d'#' -f1 | perl -wple 's/[ \t]*//
      ig'`
  mac=`echo $hostinfo | cut -d'#' -f2 | perl -wple 's/[ \t]*//ig'`

  # make new cn*-hosts & IPs
  if echo "$old_hostname"|grep -qi "^cn"; then
    if [ $rename_cn -eq 1 ]; then
      if echo "$old_hostname"|grep -qi "^cngpu"; then
                        #cngpu*
        hostnumber=`printf "%03d" $cngpu_counter`
        new_hostname="cngpu$hostnumber"

        if echo "$old_hostname"|grep -qiE "(charm|bmc)$"; then
          let "cngpu_counter += 1"
        fi
      elif echo "$old_hostname"|grep -qi "^cndev"; then
        #cndev*
        hostnumber=`printf "%03d" $cn_counter`
                                        new_hostname="cndev$hostnumber"

                                        if echo "$old_hostname"|grep -qiE
                                            "(charm|bmc)$"; then
                                                let "cn_counter += 1"
                                        fi
                  else
                          #cn*
        hostnumber=`printf "%03d" $cn_counter`
        new_hostname="cn$hostnumber"

        if echo "$old_hostname"|grep -qiE "(charm|bmc)$"; then
          let "cn_counter += 1"
        fi
                  fi
    else
      new_hostname="$old_hostname"
    fi

    increment

    infinihost=0
    if cat "hostnames.txt"|grep -qiE ""$old_hostname".?ib$"; then
      #node has infiniband
      infinihost=1
      ib_hostname=""$new_hostname"-ib"
    fi

    if echo "$old_hostname"|grep -qiE "(charm|bmc)$"; then

      if [ $rename_cn -eq 1 ]; then
        if echo "$old_hostname"|grep -qi "charm$"; then
          mgmt_hostname="$new_hostname-charm"
        else
          mgmt_hostname="$new_hostname-bmc"
        fi
      else
        mgmt_hostname="$old_hostname"
      fi

      mgmt_ip="$ip_start.$(($third_octet $mgmt_offset)).$fourth_octet"

                  if [ $print_stuff -eq 1 ]; then
        echo "$old_hostname -> $mgmt_hostname ($mac, $mgmt_ip)"
      fi

      if [ $generate_config -eq 1 ]; then
        generate_config $mgmt_hostname $mac $mgmt_ip
      fi
                else
      host_ip="$ip_start.$third_octet.$fourth_octet"
      ib_ip="$ip_start.$(($third_octet $ib_offset)).$fourth_octet"
      if [ $print_stuff -eq 1 ]; then
        echo "$old_hostname -> $new_hostname ($mac, $host_ip)"

        if [ $infinihost -eq 1 ]; then
                          echo ""$old_hostname"-ib -> $ib_hostname (
                             $ib_mac, $ib_ip)"
                          fi
      fi

      if [ $generate_config -eq 1 ]; then
        generate_config $new_hostname $mac $host_ip

        if [ $infinihost -eq 1 ]; then
          generate_config $ib_hostname $ib_mac $ib_ip $infinihost
        fi
      fi
              fi

# make new IPs for fep*-hosts
elif echo "$old_hostname"|grep -qi "^fep"; then

    if [ $rename_fep -eq 1 ]; then
      if ! echo "$old_hostname"|grep -qi "$fephost_without_number"; then
        fephost_without_number=`echo $old_hostname|sed -e 's/[0-9].*//'`
        fep_counter=0
      fi

      hostnumber=`printf "%03d" $fep_counter`
      new_hostname="$fephost_without_number$hostnumber"
    else
      new_hostname="$old_hostname"
    fi

    increment

    infinihost=0
    if cat "hostnames.txt"|grep -qiE ""$old_hostname".?ib$"; then
                          #node has infiniband
                          infinihost=1
      ib_hostname=""$new_hostname"-ib"
    fi

    if echo "$old_hostname"|grep -qiE "(charm|bmc)$"; then
      if [ $rename_fep -eq 1 ]; then
```

```
                              if echo "$old_hostname"|grep -qi "charm$";
                                  then
                                        mgmt_hostname="$new_hostname-charm"
                              else
                                        mgmt_hostname="$new_hostname-bmc"
                              fi
          let "fep_counter += 1"
        else
          mgmt_hostname="$old_hostname"
        fi

        mgmt_ip="$ip_start.$(($third_octet $mgmt_offset)).$fourth_octet"

        if [ $print_stuff -eq 1 ]; then
          echo "$old_hostname -> $mgmt_hostname ($mac, $mgmt_ip)"
        fi

        if [ $generate_config -eq 1 ]; then
                                      generate_config $mgmt_hostname $mac
                                          $mgmt_ip
                            fi
                  else
        host_ip="$ip_start.$third_octet.$fourth_octet"
                            ib_ip="$ip_start.$(($third_octet $ib_offset)).
                                $fourth_octet"

        if [ $print_stuff -eq 1 ]; then
                                      echo "$old_hostname -> $new_hostname (
                                          $mac, $host_ip)"

                                      if [ $infinihost -eq 1 ]; then
                                              echo ""$old_hostname"-ib ->
                                                  $ib_hostname ($ib_mac,
                                                  $ib_ip)"
                                      fi
                            fi

                            if [ $generate_config -eq 1 ]; then
                                      generate_config $new_hostname $mac
                                          $host_ip

                                      if [ $infinihost -eq 1 ]; then
                                              generate_config $ib_hostname
                                                  $ib_mac $ib_ip $infinihost
                                      fi
```

```
                        fi
                fi
else
  # Everything else

  new_hostname="$old_hostname"

  increment

  infinihost=0
                if cat "hostnames.txt"|grep -qiE ""$new_hostname".?ib$";
                    then
                        #node has infiniband
                        infinihost=1
                        ib_hostname=""$new_hostname"-ib"
                fi

  if echo "$old_hostname"|grep -qiE "(charm|bmc)$"; then
                        mgmt_hostname="$old_hostname"

                        mgmt_ip="$ip_start.$(($third_octet $mgmt_offset)
                            ).$fourth_octet"

                        if [ $print_stuff -eq 1 ]; then
                                echo "$old_hostname -> $mgmt_hostname (
                                    $mac, $mgmt_ip)"
                        fi

                        if [ $generate_config -eq 1 ]; then
                                generate_config $mgmt_hostname $mac
                                    $mgmt_ip
                        fi
                else
                        host_ip="$ip_start.$third_octet.$fourth_octet"
                        ib_ip="$ip_start.$(($third_octet $ib_offset)).
                            $fourth_octet"

                        if [ $print_stuff -eq 1 ]; then
                                echo "$old_hostname -> $new_hostname (
                                    $mac, $host_ip)"

                                if [ $infinihost -eq 1 ]; then
                                        echo ""$old_hostname"-ib ->
                                            $ib_hostname ($ib_mac,
                                            $ib_ip)"
                                        fi
                                fi

                                if [ $generate_config -eq 1 ]; then
                                        generate_config $new_hostname $mac
                                            $host_ip

                                        if [ $infinihost -eq 1 ]; then
                                                generate_config $ib_hostname
                                                    $ib_mac $ib_ip $infinihost
                                        fi
                                fi
                        fi
        fi
  done < $1
}

# run_code
# $1 = filename
# $2 = third octet
# $3 = fourth octet
# $4 = ib_offset
# $5 = mgmt_offset
# %6 = network_name

run_code valid-hosts-prod.txt 32 11 "- 32" "+ 32" "prod"
#run_code valid-hosts-vm.txt 52 11 "+ 0" "+ 0" "vm"

## Special
# SWITCH-MGMT
generate_bind_special valid-hosts-switch.txt "switch"

rename_cn=0
run_code valid-hosts-dev.txt 40 11 "- 32" "+ 32" "dev"
```

Listing A.1: Generate DHCP and DNS configuration-files - The script used to generate DNS and DHCP configuration-files from the CSV-file.

```
hostname "sw-core1"
max-vlans 20
time timezone 60
time daylight-time-rule Middle-Europe-and-Portugal
module 1 type J86yyA
module 2 type J86xxA
no stack
```

```
trunk 1-8 Trk2 LACP
trunk 9-16 Trk3 LACP
trunk 41-47 Trk4 LACP
trunk 17-18 Trk11 LACP
trunk 19-21 Trk27 LACP
trunk 22-24 Trk28 LACP
trunk 25-27 Trk29 LACP
trunk 28-30 Trk30 LACP
trunk 31-33 Trk31 LACP
ip routing
ip udp-bcast-forward
vlan 1
   name "DISABLED"
   forbid 34-40,48,Trk2-Trk4,Trk11,Trk27-Trk31
   no untagged 34-40,48,Trk2-Trk4,Trk11,Trk27-Trk31
   no ip address
   exit
vlan 5
   name "NOT-AUTHORIZED"
   untagged 48
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip address 10.162.61.1 255.255.255.0
   tagged 34-40,Trk2-Trk4,Trk11,Trk27-Trk31
   exit
vlan 10
   name "PROD"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.32.1 255.255.248.0
   tagged 34-40,Trk2-Trk4,Trk11,Trk27-Trk31
   ip igmp
   exit
vlan 20
   name "DEV"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.40.1 255.255.248.0
   tagged 34-40,Trk2-Trk4,Trk11,Trk27-Trk31
   ip igmp
   exit
```

```
vlan 30
   name "INFRA"
   ip address 10.162.48.1 255.255.252.0
   tagged 34-40,Trk2-Trk4,Trk11,Trk27-Trk31
   ip igmp
   exit
vlan 40
   name "VM"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.52.1 255.255.252.0
   tagged 34-40,Trk2-Trk4,Trk11,Trk27-Trk31
   ip igmp
   exit
vlan 50
   name "GW"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.56.1 255.255.255.0
   tagged 34-40,Trk2-Trk4,Trk11,Trk27-Trk31
   ip igmp
   exit
vlan 60
   name "LOGIN-PROD"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.57.1 255.255.255.0
   tagged 34-40,Trk2-Trk4,Trk11,Trk27-Trk31
   ip igmp
   exit
vlan 70
   name "LOGIN-DEV"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.58.1 255.255.255.0
   tagged 34-40,Trk2-Trk4,Trk11,Trk27-Trk31
   ip igmp
   exit
vlan 80
   name "SWITCH-MGMT"
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.60.1 255.255.255.0
   tagged 34-40,Trk2-Trk4,Trk11,Trk27-Trk31
   ip igmp
   exit
vlan 90
   name "MGMT"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.64.1 255.255.224.0
   tagged 34-40,Trk2-Trk4,Trk11,Trk27-Trk31
   ip igmp
   exit
vlan 100
   name "ADMIN"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.59.1 255.255.255.0
   tagged 34-40,Trk2-Trk4,Trk11,Trk27-Trk31
   ip igmp
   exit
vlan 200
   name "TEMP"
   tagged 34-40,Trk2-Trk4,Trk11,Trk27-Trk31
   no ip address
   exit
timesync sntp
sntp unicast
sntp 30
sntp server priority 1 10.162.50.70
ip dns domain-name "internal"
ip dns server-address priority 1 10.162.48.11
ip dns server-address priority 2 10.162.48.12
ip route 0.0.0.0 0.0.0.0 10.162.56.11
ip route 10.162.0.0 255.255.224.0 reject
ip router-id 10.162.60.254
ip multicast-routing
```

```
router ospf
   area backbone
   redistribute static
   exit
router pim
   exit
router vrrp
interface loopback 0
   ip address 10.162.60.254
   exit
snmp-server community "public"
spanning-tree
spanning-tree Trk2 path-cost 5000
spanning-tree Trk2 priority 2
spanning-tree Trk3 path-cost 5000
spanning-tree Trk3 priority 2
spanning-tree Trk4 path-cost 6000
spanning-tree Trk4 priority 4
spanning-tree Trk11 path-cost 10000
spanning-tree Trk11 priority 4
spanning-tree Trk27 path-cost 8000
spanning-tree Trk27 priority 4
spanning-tree Trk28 path-cost 8000
spanning-tree Trk28 priority 4
spanning-tree Trk29 path-cost 8000
spanning-tree Trk29 priority 4
spanning-tree Trk30 path-cost 8000
spanning-tree Trk30 priority 4
spanning-tree Trk31 path-cost 8000
spanning-tree Trk31 priority 4
spanning-tree priority 1
vlan 5
   ip ospf 10.162.61.1 area backbone
   vrrp vrid 5
      owner
      virtual-ip-address 10.162.61.1 255.255.255.0
      priority 255
      enable
      exit
   exit
vlan 10
   ip ospf 10.162.32.1 area backbone
   ip pim-dense
      ip-addr any
      exit
   vrrp vrid 10
      owner
      virtual-ip-address 10.162.32.1 255.255.248.0
      priority 255
      enable
      exit
   exit
vlan 20
   ip ospf 10.162.40.1 area backbone
   ip pim-dense
      ip-addr any
      exit
   vrrp vrid 20
      owner
      virtual-ip-address 10.162.40.1 255.255.248.0
      priority 255
      enable
      exit
   exit
vlan 30
   ip ospf 10.162.48.1 area backbone
   ip pim-dense
      ip-addr any
      exit
   vrrp vrid 30
      owner
      virtual-ip-address 10.162.48.1 255.255.252.0
      priority 255
      enable
      exit
   exit
vlan 40
   ip ospf 10.162.52.1 area backbone
   ip pim-dense
      ip-addr any
      exit
   vrrp vrid 40
      owner
      virtual-ip-address 10.162.52.1 255.255.252.0
      priority 255
      enable
      exit
   exit
vlan 50
   ip ospf 10.162.56.1 area backbone
   ip pim-dense
      ip-addr any
      exit
   vrrp vrid 50
      owner
      virtual-ip-address 10.162.56.1 255.255.255.0
      priority 255
      enable
      exit
   exit
vlan 60
   ip ospf 10.162.57.1 area backbone
   ip pim-dense
      ip-addr any
      exit
   vrrp vrid 60
      owner
      virtual-ip-address 10.162.57.1 255.255.255.0
      priority 255
      enable
      exit
   exit
vlan 70
   ip ospf 10.162.58.1 area backbone
   ip pim-dense
      ip-addr any
      exit
   vrrp vrid 70
      owner
      virtual-ip-address 10.162.58.1 255.255.255.0
      priority 255
      enable
      exit
   exit
vlan 80
   ip ospf 10.162.60.1 area backbone
   vrrp vrid 80
      owner
      virtual-ip-address 10.162.60.1 255.255.255.0
      priority 255
      enable
      exit
   exit
vlan 90
   ip ospf 10.162.64.1 area backbone
   ip pim-dense
      ip-addr any
      exit
   vrrp vrid 90
      owner
      virtual-ip-address 10.162.64.1 255.255.224.0
      priority 255
      enable
      exit
   exit
vlan 100
   ip ospf 10.162.59.1 area backbone
   ip pim-dense
      ip-addr any
      exit
   vrrp vrid 100
      owner
      virtual-ip-address 10.162.59.1 255.255.255.0
      priority 255
      enable
      exit
   exit
```

Listing A.2: Current sw-core1 config - This is the current configuration of sw-core1.

```
hostname "sw-core0"
max-vlans 20
time timezone 60
time daylight-time-rule Middle-Europe-and-Portugal
module 1 type J86yyA
module 2 type J86xxA
no stack
interface 40
   name "desk"
exit
trunk 1-8 Trk1 LACP
trunk 9-16 Trk2 LACP
trunk 17-18 Trk10 LACP
trunk 19-21 Trk20 LACP
trunk 22-24 Trk21 LACP
trunk 25-27 Trk22 LACP
trunk 28-30 Trk23 LACP
trunk 31-33 Trk24 LACP
trunk 34-36 Trk25 LACP
trunk 37-39 Trk26 LACP
```

```
ip default-gateway 10.162.60.1
ip routing
vlan 1
   name "DISABLED"
   forbid 40-48,Trk1-Trk2,Trk10,Trk20-Trk26
   no untagged 40-48,Trk1-Trk2,Trk10,Trk20-Trk26
   no ip address
   exit
vlan 5
   name "NOT-AUTHORIZED"
   untagged 41
   ip address 10.162.61.2 255.255.255.0
   tagged 42-48,Trk1-Trk2,Trk10,Trk20-Trk26
   exit
vlan 10
   name "PROD"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.32.2 255.255.248.0
   tagged 42-48,Trk1-Trk2,Trk10,Trk20-Trk26
   ip igmp
   exit
vlan 30
   name "INFRA"
   ip address 10.162.48.2 255.255.252.0
   tagged 42-48,Trk1-Trk2,Trk10,Trk20-Trk26
   ip igmp
   exit
vlan 40
   name "VM"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.52.2 255.255.252.0
   tagged 42-48,Trk1-Trk2,Trk10,Trk20-Trk26
   ip igmp
   exit
vlan 50
   name "GW"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.56.2 255.255.255.0
   tagged 42-48,Trk1-Trk2,Trk10,Trk20-Trk26
   ip igmp
   exit
vlan 60
   name "LOGIN-PROD"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.57.2 255.255.255.0
   tagged 42-48,Trk1-Trk2,Trk10,Trk20-Trk26
   ip igmp
   exit
vlan 70
   name "LOGIN-DEV"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.58.2 255.255.255.0
   tagged 42-48,Trk1-Trk2,Trk10,Trk20-Trk26
   ip igmp
   exit
vlan 80
   name "SWITCH-MGMT"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.60.2 255.255.255.0
   tagged 42-48,Trk1-Trk2,Trk10,Trk20-Trk26
   ip igmp
   exit
vlan 90
   name "MGMT"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.64.2 255.255.224.0
   tagged 42-48,Trk1-Trk2,Trk10,Trk20-Trk26
   ip igmp
   exit
```

```
vlan 100
   name "ADMIN"
   untagged 40
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.59.2 255.255.255.0
   tagged 42-48,Trk1-Trk2,Trk10,Trk20-Trk26
   ip igmp
   exit
vlan 200
   name "TEMP"
   tagged 42-48,Trk1-Trk2,Trk10,Trk20-Trk26
   no ip address
   exit
vlan 20
   name "DEV"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.40.2 255.255.248.0
   tagged 42-48,Trk1-Trk2,Trk10,Trk20-Trk26
   ip igmp
   exit
timesync sntp
sntp unicast
sntp 30
sntp server priority 1 10.162.50.70
ip dns domain-name "internal"
ip dns server-address priority 1 10.162.48.11
ip dns server-address priority 2 10.162.48.12
ip route 0.0.0.0 0.0.0.0 10.162.56.11 distance 10
ip route 10.162.0.0 255.255.224.0 reject distance 10
ip router-id 10.162.60.253
ip multicast-routing
router ospf
   area backbone
   redistribute static
   exit
router pim
   exit
router vrrp
interface loopback 0
```

```
   ip address 10.162.60.253
   exit
snmp-server community "public"
spanning-tree
spanning-tree 42 path-cost 20000
spanning-tree 43 path-cost 20000
spanning-tree 44 path-cost 20000
spanning-tree 45 path-cost 20000
spanning-tree 46 path-cost 20000
spanning-tree 47 path-cost 20000
spanning-tree 48 path-cost 20000
spanning-tree Trk1 path-cost 5000
spanning-tree Trk1 priority 3
spanning-tree Trk2 path-cost 5000
spanning-tree Trk2 priority 2
spanning-tree Trk10 path-cost 10000
spanning-tree Trk10 priority 4
spanning-tree Trk20 path-cost 8000
spanning-tree Trk20 priority 4
spanning-tree Trk21 path-cost 8000
spanning-tree Trk21 priority 4
spanning-tree Trk22 path-cost 8000
spanning-tree Trk22 priority 4
spanning-tree Trk23 path-cost 8000
spanning-tree Trk23 priority 4
spanning-tree Trk24 path-cost 8000
spanning-tree Trk24 priority 4
spanning-tree Trk25 path-cost 8000
spanning-tree Trk25 priority 4
spanning-tree Trk26 path-cost 8000
spanning-tree Trk26 priority 4
spanning-tree priority 2
vlan 5
   ip ospf 10.162.61.2 area backbone
   ip ospf 10.162.61.2 priority 2
   vrrp vrid 5
      backup
      virtual-ip-address 10.162.61.1 255.255.255.0
      priority 250
      enable
      exit
   exit
vlan 10
   ip ospf 10.162.32.2 area backbone
   ip ospf 10.162.32.2 priority 2
   vrrp vrid 10
      backup
      virtual-ip-address 10.162.32.1 255.255.248.0
      priority 250
      enable
      exit
   exit
vlan 20
   ip ospf 10.162.40.2 area backbone
   ip ospf 10.162.40.2 priority 2
   vrrp vrid 20
      backup
      virtual-ip-address 10.162.40.1 255.255.248.0
      priority 250
      enable
      exit
   exit
vlan 30
   ip ospf 10.162.48.2 area backbone
   ip ospf 10.162.48.2 priority 2
   vrrp vrid 30
      backup
      virtual-ip-address 10.162.48.1 255.255.252.0
      priority 250
      enable
      exit
   exit
vlan 40
   ip ospf 10.162.52.2 area backbone
   ip ospf 10.162.52.2 priority 2
   vrrp vrid 40
      backup
      virtual-ip-address 10.162.52.1 255.255.252.0
      priority 250
      enable
      exit
   exit
vlan 50
   ip ospf 10.162.56.2 area backbone
   ip ospf 10.162.56.2 priority 2
   vrrp vrid 50
      backup
      virtual-ip-address 10.162.56.1 255.255.255.0
      priority 250
      enable
      exit
   exit
vlan 60
   ip ospf 10.162.57.2 area backbone
   ip ospf 10.162.57.2 priority 2
   vrrp vrid 60
      backup
      virtual-ip-address 10.162.57.1 255.255.255.0
      priority 250
      enable
      exit
   exit
vlan 70
   ip ospf 10.162.58.2 area backbone
   ip ospf 10.162.58.2 priority 2
   vrrp vrid 70
      backup
      virtual-ip-address 10.162.58.1 255.255.255.0
      priority 250
      enable
      exit
   exit
vlan 80
   ip ospf 10.162.60.2 area backbone
   ip ospf 10.162.60.2 priority 2
   vrrp vrid 80
      backup
      virtual-ip-address 10.162.60.1 255.255.255.0
      priority 250
      enable
      exit
   exit
vlan 90
   ip ospf 10.162.64.2 area backbone
   ip ospf 10.162.64.2 priority 2
   vrrp vrid 90
      backup
      virtual-ip-address 10.162.64.1 255.255.224.0
      priority 250
      enable
      exit
   exit
vlan 100
   ip ospf 10.162.59.2 area backbone
   ip ospf 10.162.59.2 priority 2
   vrrp vrid 100
      backup
      virtual-ip-address 10.162.59.1 255.255.255.0
      priority 250
      enable
      exit
   exit
```

Listing A.3: Current sw-core0 config - This is the current configuration of sw-core0.

```
hostname "sw-core2"
max-vlans 20
time timezone 60
time daylight-time-rule Middle-Europe-and-Portugal
ip access-list extended "GPN-MGMT-IN"
   100 permit tcp 10.162.62.11 0.0.0.0 10.162.0.0 0.0.255.255 established
   110 permit udp 10.162.62.11 0.0.0.0 10.162.0.0 0.0.255.255
   120 permit icmp 10.162.62.11 0.0.0.0 10.162.0.0 0.0.255.255 0
   1000 deny ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
   exit
module 1 type J86yyA
module 2 type J86xxA
no stack
interface 39
   ip access-group "GPN-MGMT-IN" in
exit
trunk 1-8 Trk1 LACP
trunk 9-16 Trk3 LACP
trunk 17-18 Trk12 LACP
trunk 37-38 Trk15 LACP
trunk 19-21 Trk32 LACP
trunk 22-24 Trk33 LACP
trunk 25-27 Trk34 LACP
trunk 28-30 Trk35 LACP
trunk 31-33 Trk36 LACP
trunk 34-36 Trk37 LACP
ip default-gateway 10.162.60.1
ip routing
vlan 1
   name "DISABLED"
   forbid 39,41-48,Trk1,Trk3,Trk12,Trk15,Trk32-Trk37
   no untagged 39-48,Trk1,Trk3,Trk12,Trk15,Trk32-Trk37
   no ip address
   exit
vlan 5
```

```
   name "NOT–AUTHORIZED"
   untagged 40−43
   ip address 10.162.61.3 255.255.255.0
   tagged 44−48,Trk1,Trk3,Trk12,Trk32−Trk37
   exit
vlan 10
   name "PROD"
   ip helper−address 10.162.48.11
   ip helper−address 10.162.48.12
   ip forward−protocol udp 10.162.48.30 6666
   ip forward−protocol udp 10.162.48.31 6666
   ip address 10.162.32.3 255.255.248.0
   tagged 44−48,Trk1,Trk3,Trk12,Trk32−Trk37
   ip igmp
   exit
vlan 20
   name "DEV"
   ip helper−address 10.162.48.11
   ip helper−address 10.162.48.12
   ip forward−protocol udp 10.162.48.30 6666
   ip forward−protocol udp 10.162.48.31 6666
   ip address 10.162.40.3 255.255.248.0
   tagged 44−48,Trk1,Trk3,Trk12,Trk32−Trk37
   ip igmp
   exit
vlan 30
   name "INFRA"
   ip address 10.162.48.3 255.255.252.0
   tagged 44−48,Trk1,Trk3,Trk12,Trk32−Trk37
   ip igmp
   exit
vlan 40
   name "VM"
   ip helper−address 10.162.48.11
   ip helper−address 10.162.48.12
   ip forward−protocol udp 10.162.48.30 6666
   ip forward−protocol udp 10.162.48.31 6666
   ip address 10.162.52.3 255.255.252.0
   tagged 44−48,Trk1,Trk3,Trk12,Trk32−Trk37
   ip igmp
   exit
vlan 50
   name "GW"
   untagged Trk15
   ip helper−address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.56.3 255.255.255.0
   tagged 44-48,Trk1,Trk3,Trk12,Trk32-Trk37
   ip igmp
   exit
vlan 60
   name "LOGIN-PROD"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.57.3 255.255.255.0
   tagged 44-48,Trk1,Trk3,Trk12,Trk32-Trk37
   ip igmp
   exit
vlan 70
   name "LOGIN-DEV"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.58.3 255.255.255.0
   tagged 44-48,Trk1,Trk3,Trk12,Trk32-Trk37
   ip igmp
   exit
vlan 80
   name "SWITCH-MGMT"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.60.3 255.255.255.0
   tagged 44-48,Trk1,Trk3,Trk12,Trk32-Trk37
   ip igmp
   exit
vlan 90
   name "MGMT"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.64.3 255.255.224.0
   tagged 44-48,Trk1,Trk3,Trk12,Trk32-Trk37
   ip igmp
   exit
vlan 100
   name "ADMIN"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.59.3 255.255.255.0
   tagged 44-48,Trk1,Trk3,Trk12,Trk32-Trk37
   ip igmp
   exit
vlan 200
   name "TEMP"
   tagged 44-48,Trk1,Trk3,Trk12,Trk32-Trk37
   no ip address
   exit
vlan 120
   name "GPN-MGMT"
   untagged 39
   ip address 10.162.62.3 255.255.255.0
   exit
logging 10.162.50.72
logging severity info
timesync sntp
sntp unicast
sntp 30
sntp server priority 1 10.162.50.70
ip dns domain-name "internal"
ip dns server-address priority 1 10.162.48.11
ip dns server-address priority 2 10.162.48.12
ip route 0.0.0.0 0.0.0.0 10.162.56.11 distance 20
ip route 10.162.0.0 255.255.224.0 reject distance 20
ip router-id 10.162.60.252
ip multicast-routing
router ospf
   area backbone
   redistribute static
   exit
router pim
   exit
router vrrp
interface loopback 0
   ip address 10.162.60.252
   exit
```

```
snmp-server community "public"
spanning-tree
spanning-tree Trk1 path-cost 5000
spanning-tree Trk1 priority 3
spanning-tree Trk3 path-cost 5000
spanning-tree Trk3 priority 2
spanning-tree Trk12 path-cost 10000
spanning-tree Trk12 priority 4
spanning-tree Trk15 priority 4
spanning-tree Trk32 path-cost 8000
spanning-tree Trk32 priority 4
spanning-tree Trk33 path-cost 8000
spanning-tree Trk33 priority 4
spanning-tree Trk34 path-cost 8000
spanning-tree Trk34 priority 4
spanning-tree Trk35 path-cost 8000
spanning-tree Trk35 priority 4
spanning-tree Trk36 path-cost 8000
spanning-tree Trk36 priority 4
spanning-tree Trk37 path-cost 8000
spanning-tree Trk37 priority 4
spanning-tree priority 3
vlan 5
   ip ospf 10.162.61.3 area backbone
   ip ospf 10.162.61.3 priority 3
   vrrp vrid 5
      backup
      virtual-ip-address 10.162.61.1 255.255.255.0
      priority 240
      enable
      exit
   exit
vlan 10
   ip ospf 10.162.32.3 area backbone
   ip ospf 10.162.32.3 priority 3
   vrrp vrid 10
      backup
      virtual-ip-address 10.162.32.1 255.255.248.0
      priority 240
      enable
      exit
   exit
vlan 20
   ip ospf 10.162.40.3 area backbone
   ip ospf 10.162.40.3 priority 3
   vrrp vrid 20
      backup
      virtual-ip-address 10.162.40.1 255.255.248.0
      priority 240
      enable
      exit
   exit
vlan 30
   ip ospf 10.162.48.3 area backbone
   ip ospf 10.162.48.3 priority 3
   vrrp vrid 30
      backup
      virtual-ip-address 10.162.48.1 255.255.252.0
      priority 240
      enable
      exit
   exit
vlan 40
   ip ospf 10.162.52.3 area backbone
   ip ospf 10.162.52.3 priority 3
   vrrp vrid 40
      backup
      virtual-ip-address 10.162.52.1 255.255.252.0
      priority 240
      enable
      exit
   exit
vlan 50
   ip ospf 10.162.56.3 area backbone
   ip ospf 10.162.56.3 priority 3
   vrrp vrid 50
      backup
      virtual-ip-address 10.162.56.1 255.255.255.0
      priority 240
      enable
      exit
   exit
vlan 60
   ip ospf 10.162.57.3 area backbone
   ip ospf 10.162.57.3 priority 3
   vrrp vrid 60
      backup
      virtual-ip-address 10.162.57.1 255.255.255.0
      priority 240
      enable
      exit
   exit
vlan 70
   ip ospf 10.162.58.3 area backbone
   ip ospf 10.162.58.3 priority 3
   vrrp vrid 70
      backup
      virtual-ip-address 10.162.58.1 255.255.255.0
      priority 240
      enable
      exit
   exit
vlan 80
   ip ospf 10.162.60.3 area backbone
   ip ospf 10.162.60.3 priority 3
   vrrp vrid 80
      backup
      virtual-ip-address 10.162.60.1 255.255.255.0
      priority 240
      enable
      exit
   exit
vlan 90
   ip ospf 10.162.64.3 area backbone
   ip ospf 10.162.64.3 priority 3
   vrrp vrid 90
      backup
      virtual-ip-address 10.162.64.1 255.255.224.0
      priority 240
      enable
      exit
   exit
vlan 100
   ip ospf 10.162.59.3 area backbone
   ip ospf 10.162.59.3 priority 3
   vrrp vrid 100
      backup
      virtual-ip-address 10.162.59.1 255.255.255.0
      priority 240
      enable
      exit
   exit
vlan 120
   ip ospf 10.162.62.3 area backbone
   ip ospf 10.162.62.3 priority 3
   exit
```

Listing A.4: Current sw-core2 config - This is the current configuration of sw-core2.

```
hostname "sw-mgmt"
max-vlans 20
time timezone 60
time daylight-time-rule Middle-Europe-and-Portugal
module 1 type J86yyA
module 2 type J86xxA
no stack
trunk 1-2 Trk10 LACP
trunk 3-4 Trk11 LACP
trunk 5-6 Trk12 LACP
ip default-gateway 10.162.60.3
ip routing
ip udp-bcast-forward
vlan 1
   name "DISABLED"
   forbid 7-48,Trk10-Trk12
   no untagged 7-48,Trk10-Trk12
   no ip address
   exit
vlan 5
   name "NOT-AUTHORIZED"
   untagged 46-48
   ip address 10.162.61.4 255.255.255.0
   tagged Trk10-Trk12
   exit
vlan 10
   name "PROD"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.32.4 255.255.248.0
   tagged Trk10-Trk12
   ip igmp
   exit
vlan 20
   name "DEV"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.40.4 255.255.248.0
   tagged Trk10-Trk12
   ip igmp
   exit
vlan 30
   name "INFRA"
   ip address 10.162.48.4 255.255.252.0
   tagged Trk10-Trk12
   ip igmp
   exit
vlan 40
   name "VM"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.52.4 255.255.252.0
   tagged Trk10-Trk12
   ip igmp
   exit
vlan 50
   name "GW"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.56.4 255.255.255.0
   tagged Trk10-Trk12
   ip igmp
   exit
vlan 60
   name "LOGIN-PROD"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.57.4 255.255.255.0
   tagged Trk10-Trk12
   ip igmp
   exit
vlan 70
   name "LOGIN-DEV"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.58.4 255.255.255.0
   tagged Trk10-Trk12
   ip igmp
   exit
vlan 80
   name "SWITCH-MGMT"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.60.4 255.255.255.0
   tagged 7-48,Trk10-Trk12
   ip igmp
   exit
vlan 90
   name "MGMT"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.64.4 255.255.224.0
   tagged 7-48,Trk10-Trk12
   ip igmp
   exit
vlan 100
   name "ADMIN"
   ip helper-address 10.162.48.11
   ip helper-address 10.162.48.12
   ip forward-protocol udp 10.162.48.30 6666
   ip forward-protocol udp 10.162.48.31 6666
   ip address 10.162.59.4 255.255.255.0
   tagged Trk10-Trk12
   ip igmp
   exit
vlan 200
   name "TEMP"
   tagged Trk10-Trk12
   no ip address
   exit
timesync sntp
sntp unicast
sntp 30
sntp server priority 1 10.162.50.70
ip dns domain-name "internal"
ip dns server-address priority 1 10.162.48.11
ip dns server-address priority 2 10.162.48.12
ip route 0.0.0.0 0.0.0.0 10.162.56.11 distance 30
ip route 10.162.0.0 255.255.224.0 reject distance 30
ip router-id 10.162.60.251
ip multicast-routing
router ospf
   area backbone
   exit
router pim
   exit
router vrrp
interface loopback 0
   ip address 10.162.60.251
   exit
snmp-server community "public"
spanning-tree
spanning-tree Trk10 path-cost 10000
spanning-tree Trk10 priority 4
spanning-tree Trk11 path-cost 10000
spanning-tree Trk11 priority 4
spanning-tree Trk12 path-cost 10000
spanning-tree Trk12 priority 4
vlan 5
   ip ospf 10.162.61.4 area backbone
   ip ospf 10.162.61.4 priority 4
   vrrp vrid 5
      backup
      virtual-ip-address 10.162.61.1 255.255.255.0
      priority 200
      enable
      exit
   exit
vlan 10
   ip ospf 10.162.32.4 area backbone
   ip ospf 10.162.32.4 priority 4
   vrrp vrid 10
      backup
      virtual-ip-address 10.162.32.1 255.255.248.0
      priority 200
      enable
      exit
   exit
vlan 20
   ip ospf 10.162.40.4 area backbone
   ip ospf 10.162.40.4 priority 4
   vrrp vrid 20
      backup
      virtual-ip-address 10.162.40.1 255.255.248.0
      priority 200
      enable
      exit
   exit
vlan 30
   ip ospf 10.162.48.4 area backbone
   ip ospf 10.162.48.4 priority 4
   vrrp vrid 30
      backup
      virtual-ip-address 10.162.48.1 255.255.252.0
      priority 200
      enable
      exit
   exit
vlan 40
   ip ospf 10.162.52.4 area backbone
   ip ospf 10.162.52.4 priority 4
   vrrp vrid 40
      backup
      virtual-ip-address 10.162.52.1 255.255.252.0
      priority 200
      enable
      exit
   exit
vlan 50
   ip ospf 10.162.56.4 area backbone
   ip ospf 10.162.56.4 priority 4
   vrrp vrid 50
      backup
      virtual-ip-address 10.162.56.1 255.255.255.0
      priority 200
      enable
      exit
   exit
vlan 60
   ip ospf 10.162.57.4 area backbone
   ip ospf 10.162.57.4 priority 4
   vrrp vrid 60
      backup
      virtual-ip-address 10.162.57.1 255.255.255.0
      priority 200
      enable
      exit
   exit
vlan 70
   ip ospf 10.162.58.4 area backbone
   ip ospf 10.162.58.4 priority 4
   vrrp vrid 70
      backup
      virtual-ip-address 10.162.58.1 255.255.255.0
      priority 200
      enable
      exit
   exit
vlan 80
   ip ospf 10.162.60.4 area backbone
   ip ospf 10.162.60.4 priority 4
   vrrp vrid 80
      backup
      virtual-ip-address 10.162.60.1 255.255.255.0
      priority 200
      enable
      exit
   exit
vlan 90
   ip ospf 10.162.64.4 area backbone
   ip ospf 10.162.64.4 priority 4
   vrrp vrid 90
      backup
      virtual-ip-address 10.162.64.1 255.255.224.0
      priority 200
      enable
      exit
   exit
vlan 100
   ip ospf 10.162.59.4 area backbone
   ip ospf 10.162.59.4 priority 4
   vrrp vrid 100
      backup
      virtual-ip-address 10.162.59.1 255.255.255.0
      priority 200
      enable
      exit
   exit
```

Listing A.5: Current sw-mgmt config - This is the current configuration of sw-mgmt.