OpenBSD and Linux: Insights into a migration project at the INI

Stephan A. Rickauer

6. Mai 2007

Zusammenfassung

The Institute of Neuroinformatics (INI) at ETH / University of Zurich has always been strongly Linux focused. In fact, over 90% of our clients run Linux. However, with time we've also seen a number of problems we couldn't fix with Linux: Real redundancy on our firewalls with state synchronisation, life cycle and upgrade problems as well as reliability issues.

We evaluated OpenBSD two years ago and were surprised, if not astonished, by the most professional approaches this OS and their developers take. We started to move one machine after the other to OpenBSD and, in the meantime, migrated 80% of our server infrastructure to OpenBSD. Not only that, it turned out it is also a perfect system to virtualize.

In this talk I will vigorously show the reasons why one migrates to OpenBSD, how that actually took place and what problems we faced and still face - based on real life experience. I will not only cover technical aspects but also managment and life cycle issues.

At the end, the audience should have a better understanding of when to use which OS depending on their environment.

This is my publication about firewall HA (Linux-Magazin): http://www.ini.unizh.ch/~stephan/articles/069_linux-bsd-fw.pdf (it will also be covered in the talk)

Version 1.0

Stephan A. Rickauer (stephan.rickauer@ini.phys.ethz.ch)

OpenBSD and Linux: Insights into a migration project at the Institute of Neuroinformatics

1 Abstract

The Institute of Neuroinformatics (INI) at ETH / University of Zurich has decided to migrate most of their infrastructure services from Linux to OpenBSD. This paper sheds some light on the reasons behind this as well as the technical and management issues involved.

2 1 Introduction

IT departments of educational organisations are always challenged to build robust and reliable infrastructures on which research can rely, even if money is short. Free Software has been proven to be a cost-effective as well as a technologically solid approach to take this challenge into account.

The Institute of Neuroinformatics (INI) was established at the University of Zurich and ETH Zurich in 1995 and its information technology was based on Free Software from the very beginning, for technical and ideological reasons. It's been a logical step for us to solve further problems with Free Software as well, whenever it is possible.

3 2 The IT infrastructure at INI

Our IT infrastructure consists of roughly 120 Linux workstations, 20 servers and 30 special purpose machines as well as network devices - all maintained by one 100% position. As an institute of the University of Zurich we are physically connected to the University's network. However, due to organisational issues we've decided to separate our networks by using a clustered firewall. At the same time, we started to use our own DMZ, WLAN and LAN segments, which required us to set up all basic infrastructure services ourselves some years ago (see figure 1).

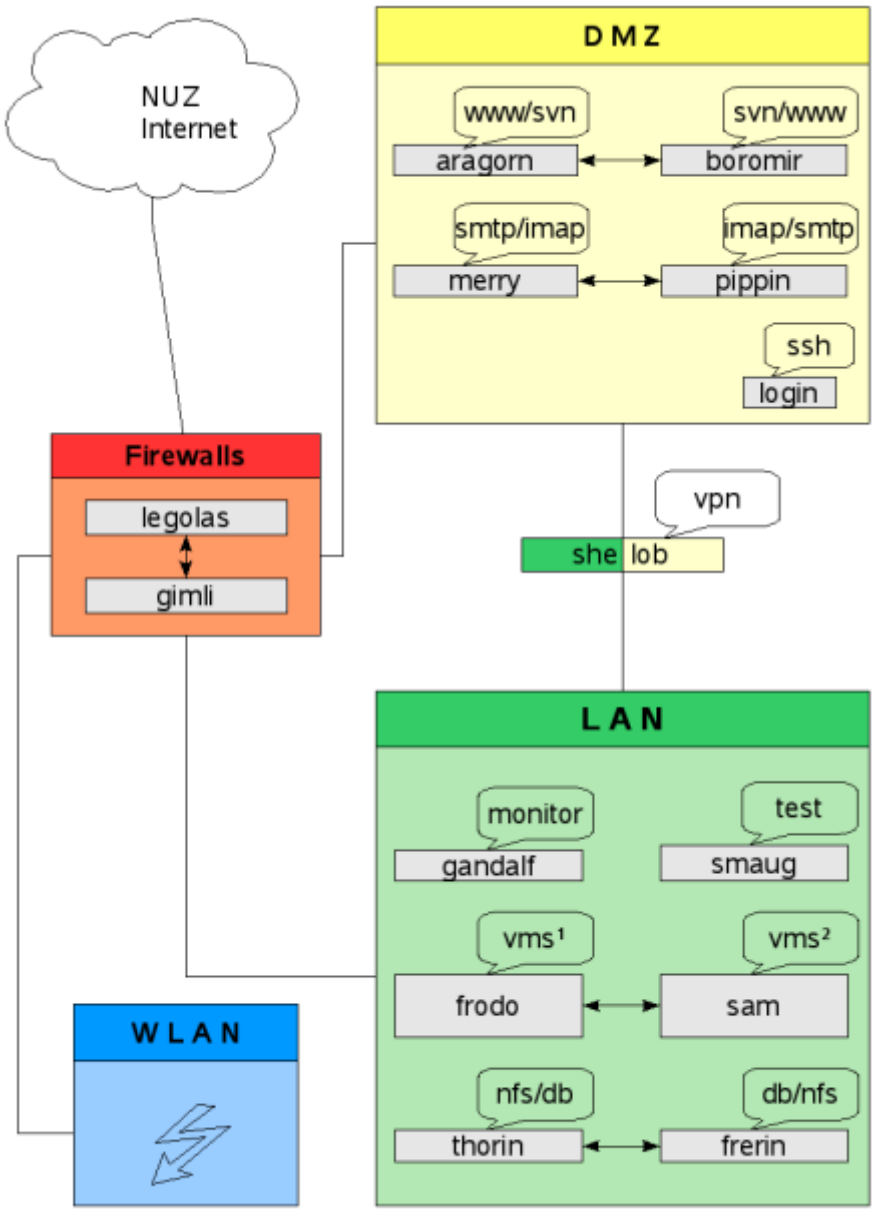


Figure 1: Schematic overview of the network of INI

3.1 2.1 Firewalls

Two i386 based machines called 'legolas' and 'gimli' running SuSE 9.0 served as firewalls in an active/passive failover cluster using 'heartbeat'. The firewalls are connected to all segments (NUZ, DMZ, WLAN, LAN). For a packet filter we used 'iptables' on the 2.4 kernel and 'fwbuilder' for managing the rule sets in a graphical user interface.

3.2 2.2 DMZ: External DNS, SMTP and IMAPs

Two i386 based machines called 'merry' and 'pippin' running SuSE 9.1 served as external DNS, SMTP and IMAPs servers. Again, these were set up using heartbeat as a failover solution in an active/passive cluster, together with 'drbd', the distributed replicating block device in order to be able to synchronize the file systems.

A dedicated machine was set up to act as a web and subversion server, running FreeBSD 4.x. Because of FreeBSD's excellent jailing techniques, this machine was also used to allow worldwide SSH logins.

3.3 2.3 LAN: File services, int. DNS, NIS, DHCP and CUPS

Two i386 based machines called 'frodo' and 'sam' running SuSE 9.0 were set up to serve as file server systems, using NFSv3 and Samba. Additionally, this active/passive cluster (heartbeat/drbd) served all DHCP client request and hosted CUPS, the Common Unix Printing System. To jail DNS queries into one network segment only, this cluster has also been set up to answer internal DNS requests and to forward recursive lookups to the external DNS servers (merry/pippin). NIS, the Network Information System, was also made available using frodo as a master and sam as a slave NIS server.

4 3 Problems with Linux infrastructure

We will now address our main problems that we've encountered using the Linux based layout detailed above.

4.1 3.1 Life Cycle Management

"Tomorrow, and tomorrow, and tomorrow, creeps in this petty pace from day to day - to the last syllable of recorded time. –Shakespeare

Our daily experience shows us the volatile nature of the material world. Within IT, the life cycle defines a period of time a vendor, a person or a community actively supports a software product, including the delivery of security and reliability patches.

The shorter a life cycle is, the more intense is the related management impact. This applies to hardware and software, and to operating system software in particular. Development cycles are so quick in the FOSS world that maintaining a secure and robust system has become a difficult task. Operating System vendors (a.k.a. Distributors) and various communities have different answers to the question of how to reconcile those apparently contrasting topics.

4.1.1 3.1.1 Community releases

Novell and Redhat support their community driven Linux distributions called 'OpenSuSE' and 'Fedora' for approximately two years. This makes these community releases unusable for professional environments, because one is forced to upgrade or reinstall all systems at least every two years. Furthermore, complex Linux distributions are known to not perform too well with upgrades, so experienced people always recommend reinstallation from scratch.

4.1.2 3.1.2 Business releases

However, Linux products which target the business market, e.g. SuSE Linux Enterprise Server or Red Hat Enterprise Linux, have a much longer life cycle of seven years. Other Linux distributions have either no defined life cycle (Debian) or start to extend their life cycles for special business releases (Ubuntu LTS, 'Long Term Support').

Life Cycle issues are almost always underestimated and can lead, if not carefully approached, to a very high workload on IT staff. A long life cycle can ease the issue of 'staying supported by a vendor' but at what price?

4.1.3 3.1.3 The price of long life cycles

Currently, the market believes in the dogma of 'the longer the life cycle, the lower the costs'. This may be true if one only takes the maintenance workload of reinstallations into account and neglects the security aspects. The software of quality focused projects becomes more secure over time [2]. This means the longer the life cycle the less secure software becomes, simply because new security technologies, for example exploit mitigation techniques, are only implemented in new software. This is true even with backports and patches made available for old releases. And because security problems can be very expensive once they have been proven to be exploitable, long life cycles can be expensive, too.

One solution to this problem we will look closer at later, can be to have a very short life cycle but to also provide an easy-peasy, fool proof and high quality upgrade mechanism.

4.2 3.2 Cluster failover

The two core components used on two of the three clusters are 'heartbeat' and 'drbd'. To understand the problems involved with this setup we have to address both technologies individually. Furthermore, restrictions of iptables in a clustered environment also need to be addressed.

4.2.1 3.2.1 heartbeat

Heartbeat, a very well known piece of software, mostly used in Linux based environments, assigns a virtual IP to the active node of a two-tier (v1) or multi-tier (v2) cluster. In case of a failure of node A, node B takes over this virtual IP and propagates a new IP to MAC relationship using ARP broadcasts, so that other devices will renew their ARP cache.

The process of broadcasting the new IP->MAC relationship is a time consuming one and usually takes around 20 seconds. Depending on the network size and the configuration of its network devices it may take even longer. During that time, neither node A or B are available (A because it's down and B because it's not reachable over the network yet).

In addition to the IP reassignment activities of heartbeat, it also manages resources and services using a script. When node A fails, a script on node B is executed which basically starts all those services which were hosted on node A before. However, since this script doesn't do magic but restarts services only, clients will need to reconnect, which is troublesome for some applications (e.g. NFS, SSL based services etc.). Open sessions are not taken into account at all.

This behaviour renders a failover concept based on heartbeat inconvenient if not unusable for us.

4.2.2 3.2.2 drbd

The Distributed Replicating Block Device (DRBD) software mirrors a whole block device over the network. One could see it as a networked RAID-1. This technology has been used in our set up on all clusters other than the firewalls. Of all clustering components it proved itself to be the most reliable. However, the version shipped with SuSE 9.0 wasn't able to do a QuickSync so a 160GB disk had to be fully resynchronised after downtime, which tool several hours. Later versions fixed this issue (shipped with SuSE >9.0).

However, DRBD introduces another layer of complexity to the system. Because it requires a kernel module, patches to one of the nodes can easily affect the entire cluster setup, especially if your favourite vendor supplies broken patches.

In our setup we made the system software, like Apache, MySQL etc. and their configuration files reside on drbd, too, using symlinks. This way one doesn't need to manually keep configuration files or software changes in sync on both machines. However, because of this design choice we could only patch base parts of the system on the *active* node, because without using a clustered file system, the file system on top of drbd can of course only be mounted read-write on one node.

4.2.3 3.2.3 netfilter/iptables

State of the art packet filters have to keep track of handled connections using a state table. If one uses more than one firewall it is crucial that this state table information is synchronised

between hosts. During our usage of iptables on kernel 2.4 it lacked the support for session synchronisation. Therefore, using heartbeat or any similar technology on our firewalls to switch from node A to B would disconnect ongoing network sessions in case of a failover. Session synchronisation became crucial for us when we started to use VPN over WLAN internally. Though connection tracking for iptables has been introduced in Linux kernel 2.6.14 it wasn't available at that time and would still have to prove its quality.

4.3 3.3 Indirect Insecurity

Due to the reliable way in which machine A failed to fail over to machine B and vice versa in our setup we've seen ourselves exposed to something I call 'indirect insecurity'. Overall OS and service security decreased because we were no longer motivated to apply security patches. Whenever we wanted to do maintenance, e.g. apply a patch, we had to make the machines fail over which often didn't work. Furthermore, with iptables and its lack of session synchronisation we would always loose active network sessions.

Both are unacceptable for any professional IT environment.

5 4 The migration to OpenBSD

Realising the above mentioned problems in 2005, our institute decided to migrate most services to OpenBSD. The following sections explain how OpenBSD is able to help fix those.

5.1 4.1 Why OpenBSD at all?

OpenBSD has not only a great reputation in regard to security but rather emphasizes high quality in all respects. Since we'd already had some experiences with FreeBSD and liked the general BSD concept, we decided to have a look into OpenBSD and see whether it would meet our expectations.

5.1.1 4.1.1 Fixed and short Life Cycle

The OpenBSD project has not only chosen a very short life cycle deliberately, but also decided to have a fixed release schedule: one release every six months (1st May and 1st November). As explained in 3.1.3, a long life cycle has to be a compromise with security and reliability, since software always improves over time. The apparently obvious conclusion that this would lead to an increased maintenance workload because of even more frequently occuring upgrades has turned out to be unfounded.

The OpenBSD project understands upgrades as a vital part of system administration as well as an important measure to keep an infrastructure secure. Therefore, a very easy way of upgrading is provided, which allows even inexperienced administrators to upgrade an entire

system within a couple of minutes. Surprisingly enough, OpenBSD's upgrade mechanism is so rock-solid that machines can be upgraded safely even without physical access.

5.1.2 4.1.2 CARP

CARP [1], the Common Address Redundancy Protocol, is a Free implementation of Cisco's VRRP and provides a means of having a virtual MAC address assigned to a set of n machines. In contrast to heartbeat, no time consuming IP to MAC repropagation is required in case of node failures since the MAC/IP relationship stays static and is therefore already known to the network (see figure 2). CARP enables us to switch from one firewall node to its standby within a second. On switch-over the backup firewall assumes the identity of the primary firewall, and continues where the latter left off. Existing connections are preserved (see 4.1.3), and network traffic continues as if nothing had happened.

Furthermore, CARP allows active/active setups in certain environments and is not limited to two nodes, but can theoretically be used with up to 256 nodes.

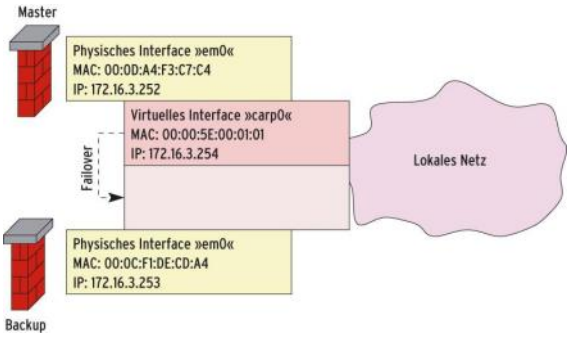
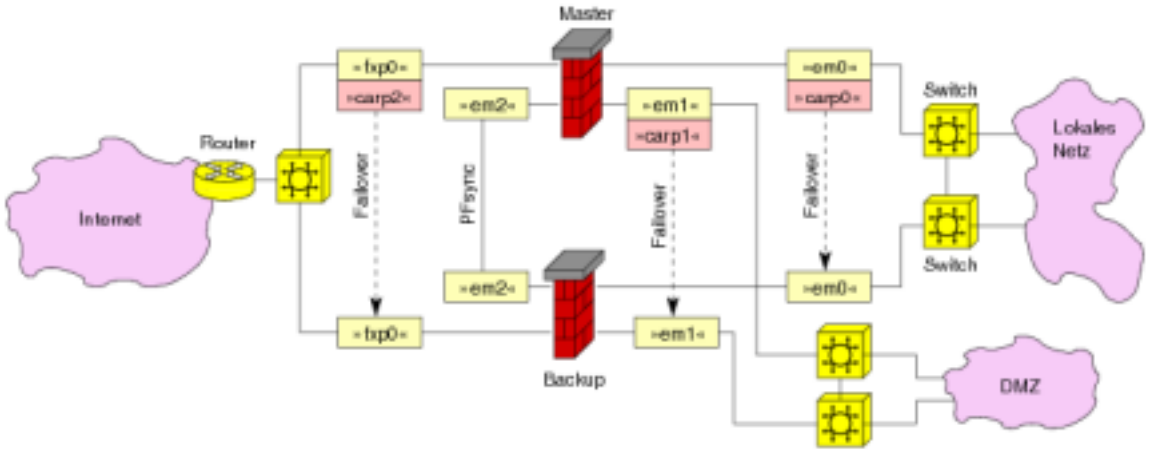


Figure 2: Principles of CARP using a Master/Backup configuration

5.1.3 4.1.3 pf and pfsync



Pf, OpenBSD's packet filter and competitor to Linux' iptables is advanced in many regards and one feature was of particular interest for us: pfsync, the packet filter state table logging interface. It enables pf to synchronize state table information to a second host. In combination with CARP one can build a highly available firewall cluster, that can fail over entirely transparently for all interfaces and connected network segments on the firewall (see figure 3).

Figure 3: CARP/pf/pfsync set up at the INI

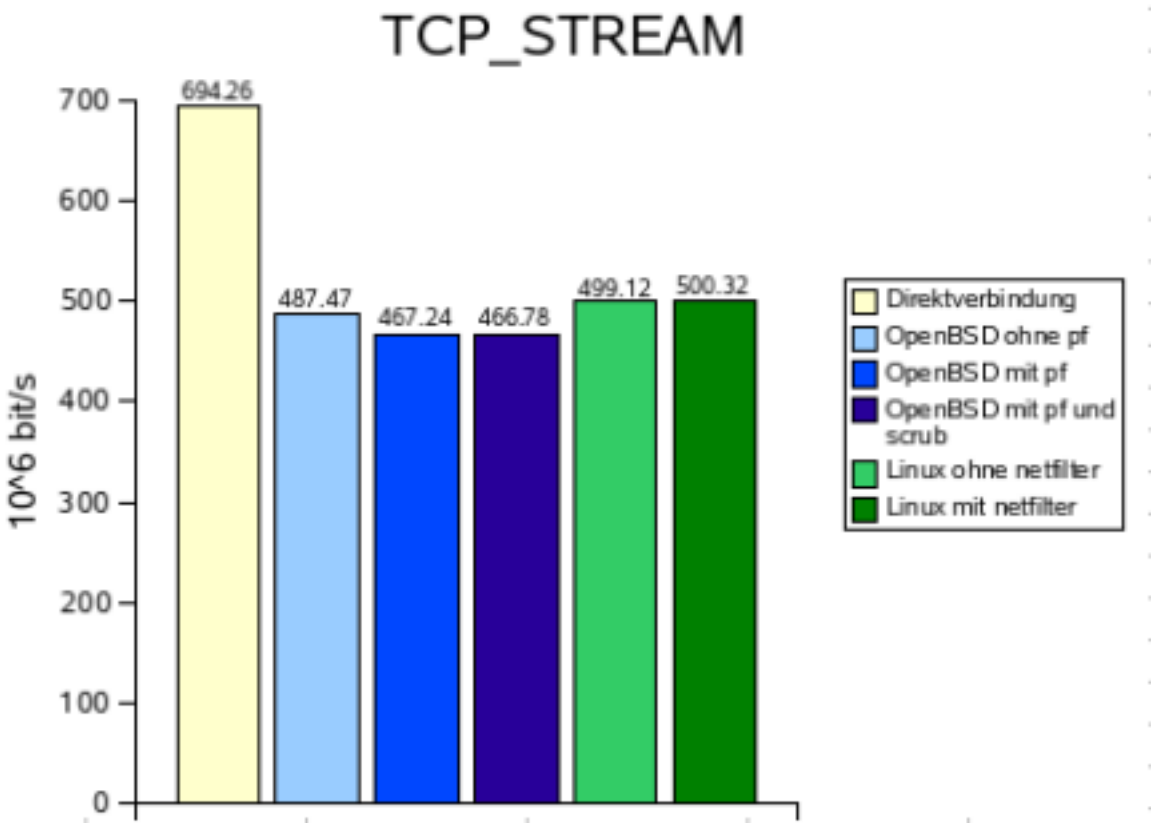
5.1.4 4.1.4 Secure by default

Some interesting guidelines drive the OpenBSD project, of which one of them is to be 'secure by default'. It is believed it doesn't make sense to require somebody to turn a security feature on. That goes along the lines of another guideline phrased into 'knobs suck'. Instead, security features should be switched on or be active by default without requiring somebody to press a button.

My favourite example is a plain chair. Not many people would consider a chair a security device. In fact, when you think about it, it is. All it does is to provide a secure means for you to maintain a specific position. It does this in a secure way and even without having you to think about it. But you will, as soon as the security feature of the chair 'is turned off' (=chair breaks).

The OpenBSD project doesn't strive for perfection by making the system unusable and calling it secure at the same time, but rather provides a most functional system, which is secure by default [3].

5.1.5 4.1.5 Performant enough



Before migrating crucial parts of an infrastructure one must make sure that no new bottlenecks are introduced. Therefore, we measured the performance of both Linux and OpenBSD systems [4], with and without packet filters enabled. Though this paper can't explain all measurements in detail, we'd like to show at least the concluding graph (figure 4), that shows that OpenBSD 3.7 is practically as fast as Linux 2.4.

Figure 4: Linux 2.4 versus OpenBSD 3.7 with netperf's TCP_STREAM

5.2 4.2 Unforeseen issues with OpenBSD

Though OpenBSD seemed to fix most of the problems we've faced so far, there were two issues attached to the migration we didn't foresee.

5.2.1 4.2.1 ftp-proxy

FTP is a firewall unfriendly protocol, since remote machines initiate connections by using random ports. In passive mode, all connections are initiated by the client, but still on random ports. A firewall would need to dynamically open and close ports depending on the current connections which becomes even more tricky when using NAT. This is where ftp-proxy comes in.

The release of OpenBSD we used initially for the firewall cluster was 3.7. This was the last release before a new and much more powerful version of the ftp-proxy was introduced. The one shipped with 3.7 had the major disadvantage that all outgoing ports needed to be opened in order to make ftp-proxy work properly. Something you wouldn't really want for a protected network. Camiel Dobbelaar, the author of the new ftp-proxy, immediately offered to provide us with a pre-compiled version, so we could use the 'better' one even with 3.7. All subsequent releases of OpenBSD did not require us to tweak ftp-proxy any longer.

5.2.2 4.2.2 OpenBSD flavours

OpenBSD has several so called flavours one can use: -current, -release and -stable. Whenever one wants to follow -stable to get all patches automatically, one has to rebuild the entire system from source. What sounds like a nightmare has, once again, been designed in such a clean way that it actually wasn't a real problem in the end. However, since the concept of rebuilding the entire system from source was new to us, it did require us to learn how to achieve this. Later, we learned how to minimize the involved work by using 'release' (see 6.5).

5.3 4.3 Why we didn't use OpenBSD everywhere

Though our main problems could be solved by OpenBSD there are still services we cannot easily migrate because of a lack of support.

5.3.1 4.3.1 iSCSI

Our Institute based its storage virtualisation concept on iSCSI which enables us to simply assign virtual devices to iSCSI capable machines. For us, this is a very cost-effective solution in comparison to fibre channel based setups and performant enough. Since OpenBSD has neither a software iSCSI implementation nor supports iSCSI host bus adapters (HBA) we currently can't move those servers which require lots of storage to OpenBSD. This affects our IMAP as well as our NFS/Samba server.

5.3.2 4.3.2 TSM Backup

The University of Zurich offers all of its institutes a cheap and very convenient interface to an IBM Tivoli Storage Manager (TSM). Using the TSM client software we can easily backup and archive terabytes of data without being forced to deal with capacities, tape drive issues etc. ourselves. However, there is no TSM client for OpenBSD offered by IBM. Attempts to make the Linux version run in an emulated way may succeed but would base a crucial part of our infrastructure on a very fragile foundation. Since we only need a TSM client on our IMAP and NFS/SMB servers anyway (see 4.3.1), we can currently do without it on OpenBSD.

5.3.3 4.3.3 LVM and file system

Managing multiple terabytes of storage is a challange for classic UNIX systems. Devices with static sizes are not flexible enough. Logical Volume managers (LVM) try to fix this by introducing an abstraction layer to raw devices. OpenBSD currently lacks a Logical Volume Manager so we can't use OpenBSD on our storage demanding systems (again, IMAP and NFS/Samba servers).

Furthermore, we require a file system that enables growth while being mounted. Currently, UFS, the default file system on OpenBSD, does not support this feature, also because there is no underlying technology that would enable the device to grow in the first place.

On those machines in question we use SLES10 with XFS.

6 5 The outcome / Summary

As a summary, table 1 provides an overview of which systems we have migrated or are about to migrate to OpenBSD or SLES10, given the reasons and explanations above.

Machine name	Purpose / Service	Former OS	Current OS
Legolas	Firewall	SuSE 9.0	OpenBSD 4.0-stable
Gimli	Firewall	SuSE 9.0	OpenBSD 4.0-stable
Merry	SMTP/ext. DNS	SuSE 9.1	OpenBSD 4.0-stable*
Pippin	IMAPs	SuSE 9.1	SLES 10
Frodo	NFS/Samba	SuSE 9.0	SLES 10
Sam	NFS/Samba	SuSE 9.0	SLES 10
Balin	Int. DNS/DHCP	SuSE 9.0	OpenBSD 4.0-stable
Dwalin	Int. DNS/DHCP	SuSE 9.0	OpenBSD 4.0-stable
Bombadil	CUPS	SuSE 9.0	OpenBSD 4.0-current
Arwen	Subversion	FreeBSD 4.x	OpenBSD 4.0-current
Kobur	RPM/apt-get server	n/a	OpenBSD 3.9-release
Elrond	Build host	n/a	OpenBSD 4.0-stable
Soekris	SSH login	FreeBSD 4.x	OpenBSD 4.0-stable

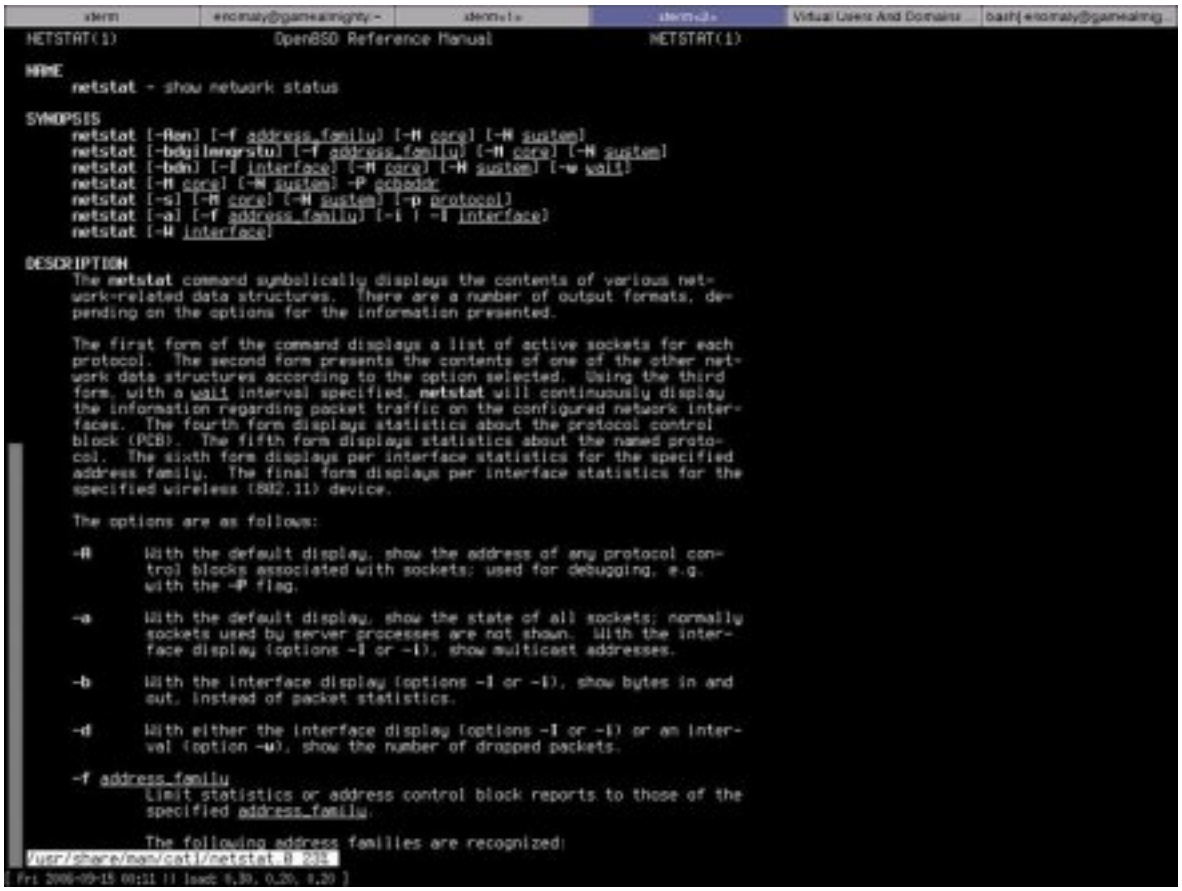
Table 1: Migration overview * migration not yet performed

7 6 Post migrational issues

Once our firewalls, DNS, DHCP and CUPS servers were migrated we of course become more experienced with OpenBSD and discovered features or general issues we were positively surprised with.

7.1 6.1 High quality of documentation (RTFM)

Since we mainly used Linux before we weren't used to look at the man pages at all. After being told on misc@openbsd.org, one of OpenBSD's mailing list, more than once to look at the man pages we realised that the OpenBSD project emphasizes documentation differently than others. Not having something documented is effectively treated simply as a bug that needs to be fixed. This leads to a very high quality of documentation, which is a great help for everbody, as shown in figure 5.



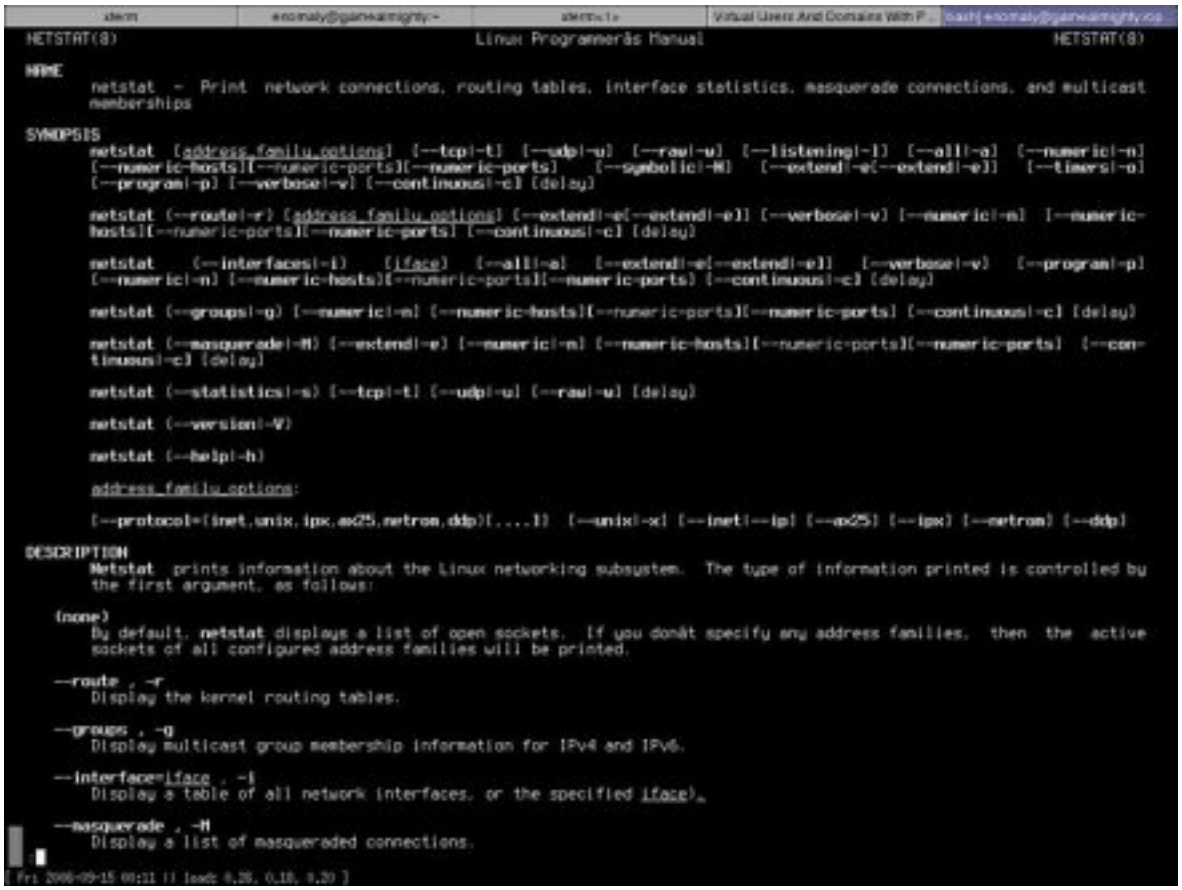


Figure 5: Visual representation of the netstat man page (l: Linux, r: OpenBSD)

7.2 6.2 High quality of community support

In the process of learning, asking lots of questions is naturally. In contrast to commercial or conventional software, people sometimes, when asked about the downsides of Free Software, mention the missing vendor support. While this may be true for some projects it definitely isn't for OpenBSD. All of our questions where answered on the community channels and mostly surprisingly quickly. In addition to this there are also several established companies which would offer commercial support for OpenBSD.

7.3 6.3 Readability of pf rules

Because of the growing number of firewall rules we already had (>1000) and the fact that iptables' syntax isn't very easy to read for humans we were used to administering our rule set with FWBuilder, a graphical, object-oriented management tool. Our rule set migration from iptables to pf was greatly supported by the fact that FWBuilder is able to generate iptables as well as pf rule sets. We currently still use FWBuilder, mostly for historical reasons. However, since pf's rule sets are so easily readable one can simply do without a graphical interface.

7.4 6.4 max-src-conn-rate

A feature we discovered months later, after the first deployment of an OpenBSD machine, is pf's 'max-src-conn-rate'. It allows the rate of new connections to be limited. The connection rate is an approximation calculated as a moving average. This allows effective countermeasures for all sorts of DoS attacks. In fact, since we have enabled max-src-conn-rate on our firewalls all regular brute force attack attempts on SSH services have vanished.

7.5 6.5 release(8)

Patching the systems with source code diffs and recompiling the entire system is acceptable if you only have a couple of machines. However, we now have a number of OpenBSD systems and patching them individually would consume too many resources. For this purpose, the OpenBSD project enables you to make your own 'release'. Once you have a dedicated build machine you can easily prepare a new release yourself and deploy it in a couple of minutes. This is a very powerful measure for dealing with a very large number of installations.

7.6 6.6 Efficient use of system resources

We were astonished how efficiently OpenBSD makes use of system resources, e.g. CPU and RAM. It's resource usage is actually so modest that we currently also use it on VMWare virtual machines with only 64MB each. Our SNMP graphs demonstrate the switch from Linux to OpenBSD on our firewalls quite dramatically. We introduced OpenBSD in September and turned on OpenBSD's power saving mode in May. In summary, OpenBSD utilizes the same amount of CPU cycles when in power save mode as Linux 2.4 did without (see figure 6).

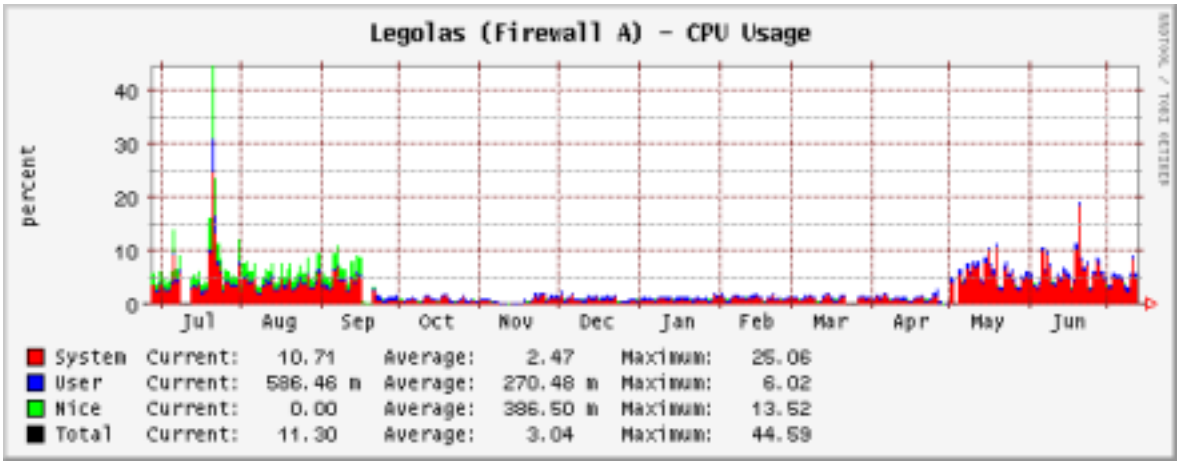


Figure 6: CPU usage of Linux (July-September) versus OpenBSD (Sept.-June)

8 7 Future ideas

The current development of OpenBSD holds exiting possibilities for us. Even more enterprise class tools are included in the current base system, e.g. hoststated and ifstated.

8.1 7.1 Hoststated

hoststated is the host status daemon for server load balancing. It has been introduced in OpenBSD 4.1. Its main purpose is to keep pf(4) tables up to date as well as any related pf redirect rules. To communicate with pf, hoststated uses its anchor facility. This feature will enable us to set up monitored load-balanced services which are controlled by pf. It may be seen as a sophisticated replacement for commercial load balancing products.

8.2 7.2 Ifstated

The ifstated daemon runs commands in response to network state changes, which it determines by monitoring interface link state or running external tests. For example, it can be used with CARP to change running services or to ensure that CARP interfaces stay in sync, or with pf to test server or link availability and modify translation or routing rules. This feature allows us to bind services to specific CARP states. It therefore can act as a clean replacement for heartbeat's resource management facilities.

9 8 Conclusion

The original project to migrate two firewalls to OpenBSD has led in the end to eight of our Linux based services being migrated. Furthermore, three new OpenBSD based servers have been introduced and there are more to come.

Not only could the existing functionality of the former setup be replaced, but rather enhanced in many aspects. Firewall redundancy is now completely transparent to the user thanks to CARP/pfsync. Specific as well as overall security was increased by using pf's features such as 'max-src-conn-rate'.

However, due to a lack of iSCSI drivers and an LVM implementation we could not use OpenBSD for our file and imap servers (so far).

OpenBSD has become the strategic server platform number one in our institute.

10 9 Acknowledgements

I'd like to thank the following people for advice, comments and feedback: Adrian Whatley, Ryan McBride, Marc Balmer, Camiel Dobbelaar. In addition I'd also like to express my gratitude to the OpenBSD community which has always been a helpful and inspiring source.

11 10 References

[1] Firewall Failover with pfsync and CARP, Ryan McBride (mcbride@openbsd.org), http://www.count
carp/

[2] Milk or Wine: Does Software Security Improve with Age? Andy Ozment, Stuart E. Schlechter

[3] Exploit Mitigation Techniques Theo de Raadt http://www.auug.org.au/events/2004/auug2004/th

[4] Wehrhaft abtauchen, HA-fähige Firewall mit OpenBSD/PF Stephan A. Rickauer Linux-Magazin 2005/12