

axxeo GmbH
Kniestraße 27
30167 Hannover

Dokumentation der betrieblichen Projektarbeit

Split-Access-Routing mit Priorisierung auf Linux-Basis

Oluf Lorenzen

Oktober 2009

betreut durch Dipl.-Inform. Andreas Godzina

Ausbildungsberuf: Fachinformatiker Systemintegration

Durchführungszeitraum: 12.10.2009 – 20.10.2009

Inhaltsverzeichnis

Inhaltsverzeichnis	I
1. Einleitung	1
1.1. Umfeld der Installation und Ist-Zustand	1
1.2. Beschreibung Soll-Zustand	1
1.3. Zeitplanung	2
2. Planung	3
2.1. Ressourcenplanung	3
2.2. Split-Access-Routing	4
2.3. Priorisierung	4
3. Durchführung	8
3.1. Installation	8
3.2. Tests	8
4. Fazit	10
4.1. Kosten	10
4.2. Erweiterbarkeit	10
4.3. Alternativen	11
A. Quellen	12
B. Glossar	13
C. Akronyme	14
D. Abbildungsverzeichnis	15
E. Verzeichnis der Listings	16

1. Einleitung

1.1. Umfeld der Installation und Ist-Zustand

Die axxeo GmbH ist als Linux- und Netzwerkdienstleister 2004 in Hannover gegründet worden. Der Kerndienstleistungsbereich umfasst größtenteils *second-* und *third-Level-Support* an Debian-Linux-Servern, Planung und Bereitstellung von Firewall- und Serversystemen sowie deren Wartung. Das Projekt „*Split-Access-Routing mit Priorisierung auf Linux-Basis*“ ist ein In-House-Projekt, welches nach erfolgreicher Implementierung im Produktivnetz unter anderem für Schulungszwecke und zum Sammeln von Erfahrung eingesetzt werden soll. Darüber hinaus wird nach einiger Zeit die Funktionalität auch in eigene Produkte übernommen.

Beschreibung Ist-Zustand

Derzeit ist die axxeo GmbH über einen einzelnen ISP¹ mit 2000 Kibit/s an das Internet angebunden (Siehe Abb. 2.1 auf Seite 3, „alte Verbindung“). Diese Leitung wird firmenintern von drei bis vier Personen und innerhalb eines untervermieteten Büros von weiteren zwei bis drei Personen genutzt. Außerdem wird die Leitung zur Anbindung eines Mailservers und zum Download nächtlicher, inkrementeller Backups von Kundenservern genutzt. Für die tägliche Arbeit ist seitens der axxeo GmbH der Zugriff via SSH² auf entfernte Maschinen, sowie die Nutzung der Protokolle HTTP³ und SMTP⁴ essentiell.

Durch die Unter Vermietung hat sich die Auslastung der Internetleitung geändert. So war zuvor selten eine höhrere Latenz durch starke Auslastung festzustellen – nun ist eine volle Auslastung häufig gegeben, was die Support-Arbeit via SSH stark erschwert und verlangsamt.

1.2. Beschreibung Soll-Zustand

Um die Bandbreite zu priorisieren und die Last generell auf eine zweite Leitung aufzuteilen, wurde zunächst das Tool tc(8) und das LARTC-Howto (Hubert u.a., 2005) zur Problemlösung favorisiert. Diese Vorauswahl basiert auf einer generellen Präferenz der axxeo GmbH für Lösungen, die sich mit Debian-Bordmitteln und nach Debian-Regeln erreichen

¹Internet Service Provider / Internetdienstanbieter

²Secure Shell, sicherer Ersatz für *telnet*

³Hypertext Transfer Protocol, Netzwerkprotokoll zur Übertragung von Webseiten

⁴Simple Mail Transfer Protocol, Netzwerkprotokoll zur Übertragung von E-Mails

lassen. Die Arbeit via SSH muss trotz hoher Auslastung der Bandbreite latenzfrei möglich sein.

1.3. Zeitplanung

geplante Tätigkeit	geplante Dauer
Basiskonzept, Information, etc.	3-4 h
Installation, Konfiguration, Einbinden in bestehende Scripte	5 h
Anpassung von Parametern in Absprache mit Auftraggeber	1 h
Test des Aufbaus, Zusammenspiel von vorhandenen Systemscripten mit neuer Konfiguration bei Reboot, Anpassung etc.	4 h
Erstellung von Scripten um Leitungs-Ausfall festzustellen	3 h
Laufzeit-Tests (Last-Verteilung, Priorisierung), Simulation von versch. Ausfällen (Netzwerk-Interface, Next-Hop-Router, ...)	7 h
Dokumentation	8 h
variable Pufferzeit	3 h

Tabelle 1.1.: Planung der benötigten Zeit innerhalb des vorgeschriebenen Zeitrahmens von 35 h

2. Planung

Es wurde zunächst ein Netzplan erstellt (Abb. 2.1), der der besseren Übersicht und Darstellbarkeit halber ab dem ersten Router nach `firewall.office.axxeo.de` stark vereinfacht und auf einen weiteren Hop beschränkt ist. In der Realität folgen nach Router eins und Router zwei noch weitere Hops. Für alle Netze außerhalb von `firewall.office.axxeo.de` wurden private Netzbereiche gewählt, um nicht mit bereits vergebenen, öffentlichen IP¹-Adressen in Berührung zu kommen.

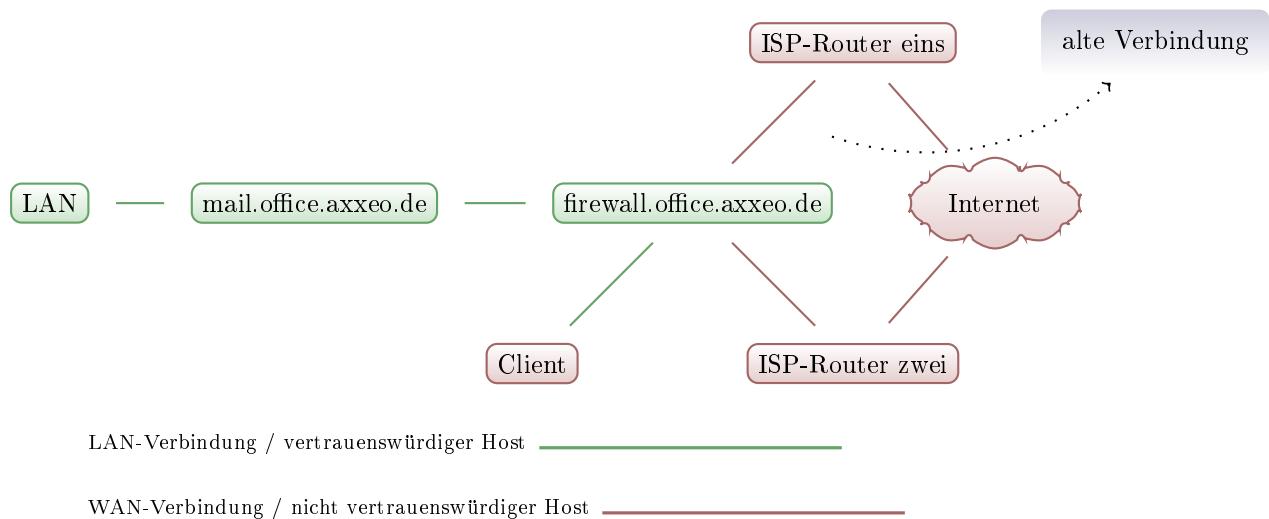


Abbildung 2.1.: Netzplan

2.1. Ressourcenplanung

Für die Umsetzung des Projekts sind keine Anschaffungen oder speziellen Einplanungen nötig; die vorhandene Firewall `firewall.office.axxeo.de` (Abb. 2.1) wird erweitert und für Simulationen steht eine gemeinsam genutzte VMWare-Workstation bereit. Zur Erarbeitung wird ein bereits zugewiesener Standard-PC-Arbeitsplatz verwendet.

¹Internet Protocol

2.2. Split-Access-Routing

Um als ersten Schritt ein überhaupt laufendes Setup mit S-A-R² zu erzeugen, wurde der im LARTC beschriebene Aufbau übernommen und den Umständen angepasst:

1. Erstellen von einer Routing-Tabelle für jedes externe Interface:

```
echo -e "200 one\n 201 two" >> /etc/iproute2/rt_table
```

2. Hinzufügen von zwei Regeln in die Routing-Policy-Datenbank, um Antworten auf Anfragen aus den jeweils anliegenden Netzbereichen (10.1.0.0/24 und 10.2.0.0/24) auch wieder auf dem passenden Interface zu versenden:

```
ip rule add from 10.1.0.0/24 table one
ip rule add from 10.2.0.0/24 table two
```

3. Hinzufügen von Routen in die im ersten Schritt erstellten Tabellen; ohne diese Einträge würden die im zweiten Schritt aufgegriffenen Pakete die jeweilige Routing-Tabelle durchlaufen, auf keinen Eintrag passen und dann wieder die main-Tabelle passieren:

```
ip r r 127.0.0.0/8 dev lo table one
ip r r 10.2.0.0/24 dev eth2 src 10.2.0.1 table one
ip r r default dev eth1 table one
ip r r 127.0.0.0/8 dev lo table one
ip r r 10.1.0.0/24 dev eth1 src 10.1.0.1 table two
ip r r default dev eth2 table two
```

4. Ersetzen der default-Route mit einem eins-zu-eins-Balancing:

```
ip r r default scope global nexthop via 10.1.0.2 ↵
    ↵dev eth1 weight 1 nexthop via 10.2.0.2 dev eth2 weight 1
```

2.3. Priorisierung

Zum Verständnis von Priorisierung und Traffic-Shaping generell wurde wieder das LARTC (Hubert u.a., 2005) genutzt. Nach der Lektüre des Kapitels 9 war ersichtlich, wie die Priorisierung funktioniert und, dass die Syntax von tc(8) selbst weder leicht zu verstehen, noch leicht zu schreiben ist. Nach weiterer Recherche³ wurde ein Howto für das Tool tcng(1) (Brown, 2006b) gefunden. Auch hier wurde nah am Beispiel gearbeitet und die Konfiguration nur wenig verändert. Die tcng(1)-Scripte für die Schnittstellen eth1 und eth2 auf `firewall.office.axxeo.de` sind unter E.7 auf Seite 19 und E.8 auf Seite 20 angehängt.

²Split-Access-Routing, Internetzugang über mehrere Anbindungen gleichzeitig

³u.a. <http://tldp.org/HOWTO/HOWTO-INDEX/networking.html>

Die Einteilung der Bandbreite für jedes Interface hat sich dabei folgendermaßen ergeben: An eth1 steht die SDSL⁴-Leitung mit 2000/2000 Kibit/s (Zeile 24 auf Seite 20) zur Verfügung, über welche Fernwartung via SSH erfolgen muss, da nur die auf diesem Interface liegende IP auf den Maschinen der Kunden freigeschaltet ist. An eth2 steht eine VDSL2+⁵-Leitung mit 50000/10000 Kibit/s (Zeile 24 auf Seite 20) zur Verfügung, welche von einem weiteren Mieter im Hause betrieben wird und auf Fair-Use-Basis genutzt werden kann.

Der Aufbau der tcng(1)-Scripte ist hierarchisch und recht einfach; erklärt wird folgend der Aufbau des Scripts für eth1.

Als erstes wird das Interface definiert (Zeile 1 auf Seite 19), für das tc(8)-Regeln erstellt werden sollen, danach wird die egress- oder ingress-qdisc gewählt (Zeile 2), in der wiederum Definitionen für Traffic-Gruppen (Zeile 3, 12, 17) und die darauf angewendeten Regeln definiert werden (Zeile 26, 28, 29). Verwendet wird allerdings nur die egress-qdisc, da nur die Datenströme wirklich beeinflusst werden können, die von der Maschine selber ausgehen. Für alle Datenströme, die möglichst ohne Verzögerung übertragen werden sollen, sind die Klassen `$interactive_*` vorgesehen (Zeile 26, 28). Alle anderen Datenströme werden über die Klasse `$other` (Zeile 17) abgewickelt.

Alle Regeln sind innerhalb des HTB⁶ (Zeile 21) definiert. Mit der Wahl des HTB wird versucht, einen einfachen, durchschaubaren⁷ Einstieg zu finden, der laut Entwickler auch genau zum Anwendungsfall passt⁸. Generell erlaubt der hierarchische Aufbau eine feingranulare Aufteilung der Bandbreite nach vielen Parametern.

Für ein besseres Verständnis ist der implementierte Aufbau unter Abb. 2.4 auf Seite 7 auch als Grafik zu sehen.

Detail-Analyse der tcng-Scripte (Listing E.7 auf Seite 19 und E.8 auf Seite 20)

Die folgenden Ausführungen wurde zu einem gewissen Teil von docum.org (Coene, 2005) übersetzt, da die Formulierungen sehr treffend sind.

rate die garantierte Bandbreite einer Klasse

ceil die maximale Bandbreite, die eine Klasse nutzen kann

⁴Symmetric Digital Subscriber Line

⁵Extended bandwidth Asymmetric Digital Subscriber Line 2

⁶Hierarchical Token Bucket packet scheduler – eine Variante des *Token-Bucket-Filter*

⁷„As said before, CBQ is the most complex qdisc available, the most hyped, the least understood, and probably the trickiest one to get right“ (Hubert u. a., 2005, Kapitel 9.5.4.)

⁸„One requirement for link-sharing is to share bandwidth on a link between multiple organizations, where each organization wants to receive a guaranteed share of the link bandwidth during congestion, but where bandwidth that is not being used by one organization should be available to other organizations sharing the link.“ (Balliache, 2003, Kapitel 2.8)

Den Parametern *rate/burst* und *ceil/cburst* werden sog. *token buckets* zugewiesen. Der *rate-bucket* wird mit *token* und der *ceil-bucket* mit *ctoken* gefüllt - es gibt also für jede Klasse zwei buckets. Die Geschwindigkeit mit der die Tolken gefüllt werden wird über *burst* und *cburst* festgelegt.

Ein Beispiel mit $rate=100$ B/s, $burst=300$ B und einem Datenversand von 200 B/s:

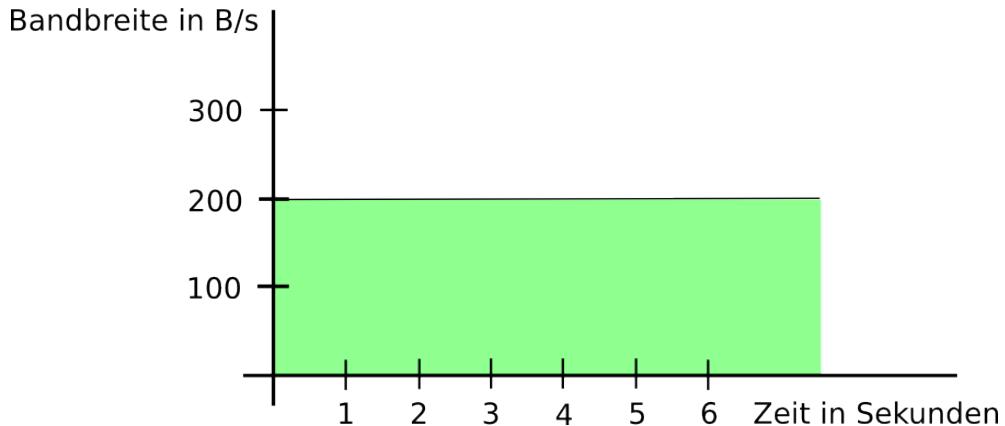


Abbildung 2.2.: gewünschte Bandbreite

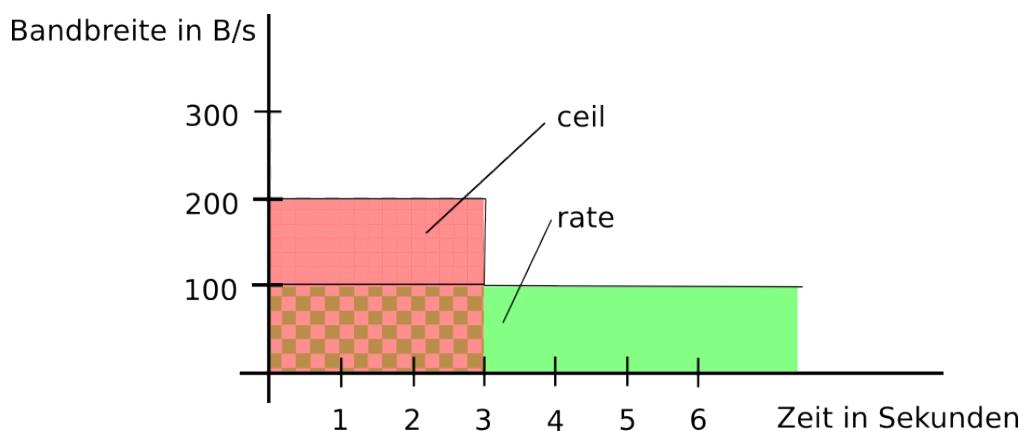


Abbildung 2.3.: Bandbreite nach HTB

- **0 - 3 Sekunden:** Datenversand kann in voller Geschwindigkeit erfolgen, es werden nachlaufende token und token aus dem bucket genutzt
- **4 - ... Sekunden:** bucket ist leer, es werden nachlaufende token genutzt

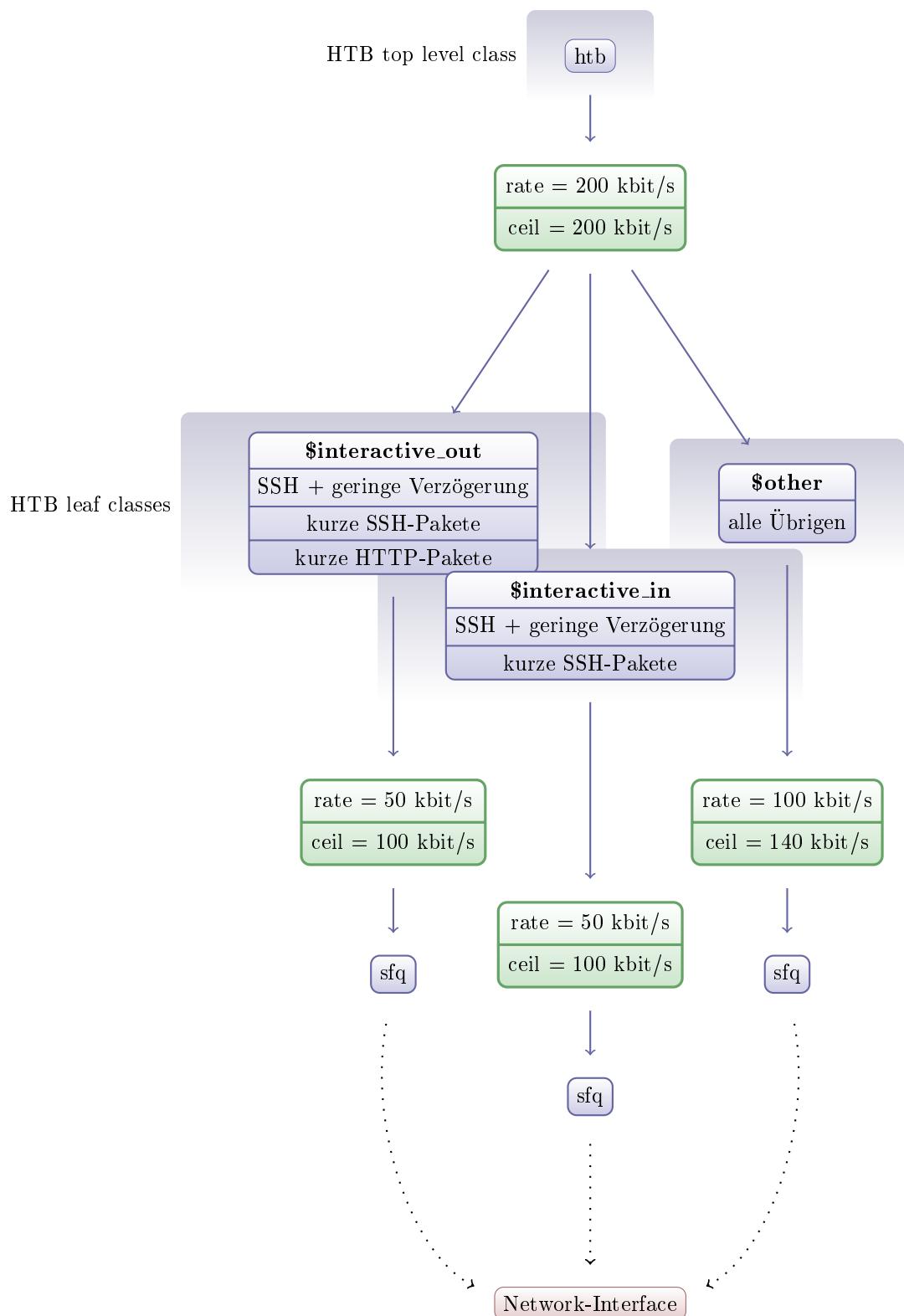


Abbildung 2.4.: grafische Darstellung des TCC-Scripts von eth1 (siehe E.7 auf Seite 19)

3. Durchführung

3.1. Installation

Installation benötigter Software/Änderung vorhandener Dateien

aptitude install tcng – Installation von tcng(1)

echo -e ”200 one\n 201 two” >> /etc/iproute2/rt_table – Hinzufügen von zwei Routing-Tabellen

Einspielen angefertigter Dateien

/opt/etc/extra-routes – siehe E.4 auf Seite 17,

bei Firewall-Start ausgeführtes Script *a*) zur Erstellung von Routing-Tabellen für beide Internet-Anbindungen; *b*) zum Füllen dieser Tabellen; *c*) zum Setzen von speziellen Routen.

/etc/init.d/traffic-shaper – siehe E.5 auf Seite 18,

nach Firewall-Start Ausgeführtes Script zum Complilen von E.7 auf Seite 19 und E.8, sowie Ausführen der ausgegebenen tc(8)-Befehle

/usr/local/etc/tc-routing.sh – siehe E.6 auf Seite 19,

Konfiguration für /etc/init.d/traffic-shaper und /opt/etc/extra-routes

/usr/local/etc/tc/*.tcc – siehe E.7 auf Seite 19 und E.8,

tcng(1)-Scripte welche in /etc/init.d/traffic-shaper genutzt werden

3.2. Tests

Die Routing-Tabellen von `firewall.office.axxeo.de` wurden wie erwartet durch die Scripte von E.1 auf Seite 16 nach E.2 und die Konfiguration der Kernel-Traffic-Kontrolle (E.3) erweitert.

- S-A-R – Das S-A-R funktioniert wie erwartet. Es wurde ein Ping von LAN an diverse externe IP-Adressen durchgeführt. Überprüft wurde die Verteilung mit dem Befehl `for i in 1 2; do ip route show cache|grep`

`-c "dev eth$i" ; done`, welcher den Routing-Cache ausliest und zählt, wieviele Routen für jedes Interface vorhanden sind. Nach Trennen einer Verbindung zum Internet an `firewall.office.axxeo.de` wurden die Routen im Routing-Cache weiterhin nach Gewichtung (Zeile 28 auf Seite 19) verteilt. Die Verbindungen über das Interface ohne Verbindung konnten nicht aufgebaut werden.

Priorisierung – Zum Test der Priorisierung wurde die default-Route auf die schmalbandige Leitung umgestellt und mit hping auf einem LAN-Client eine variierender Datenstrom zum Internet und umgekehrt erzeugt. Bei beiden Tests war die Arbeit mit SSH latenzfrei möglich.

Bei der Erzeugung eines weiteren Datenstroms von einem anderen Client im LAN wurde die Bandbreite gleichmäßig aufgeteilt.

Die Scripte zur Überprüfung des Leitungsausfalls wurden nicht erstellt, da es nicht zuverlässig möglich ist festzustellen, ob eine Leitung ausgefallen ist oder nicht. Hier wären auch sehr viele Parameter zu beachten: u. a. das Netzwerk-Interface selber, das Netzwerkkabel, der nächste Router etc. Zu überprüfen wäre dann, ob die benötigte Funktion oder nur eine zur Überwachung genutzte Komponente des jeweiligen Teils eingeschränkt funktioniert. Ein Beispiel wären hier Core-Router in einem Rechenzentrum, welche oft aus Performance-Gründen Ping-Anfragen an sich selber verwerfen.

Nach Rücksprache mit dem Auftraggeber wurde diese Funktionalität aus dem Projekt entfernt.

4. Fazit

Das Projekt konnte im vorgesehenen Zeitrahmen abgeschlossen werden (siehe Tabelle 4.2 auf der nächsten Seite) und verlief zu großen Teilen erfolgreich. Da die Vermutung, dass das S-A-R einen Ausfall einer Leitung von selbst bemerken würde, falsch war, konnte kein automatisiertes Umschalten auf eine andere Leitung implementiert werden. Dies hängt auch damit zusammen, dass viele Faktoren einen Ausfall verursachen können, welche auf unterschiedliche Weise geprüft werden müssen. Eine Ausarbeitung dessen hätte den Zeitrahmen gesprengt.

4.1. Kosten

Beschreibung	Anzahl	Einzelpreis	Gesamtpreis
Auszubildender (Planung und Durchführung)	35 Stunden	38 €	1330 €
Projektbetreuer (Gespräche und Abnahme)	2 Stunden	38 €	76 €
Gesamt			1406 €

Tabelle 4.1.: Projektkosten

Der finanzielle Nutzen des Projekts liegt mehrfach über den Kosten, da *a*) der Wegfall der Latenz bei der Support-Arbeit eine Erleichterung darstellt, welche die Arbeitsqualität erhöht; *b*) dadurch die Support-Arbeit verkürzt wird, was die Kosten für Kunden herunter setzt, was wiederum auch die vom Kunden wahrgenommene Qualität erhöht; *c*) das Feature der Priorisierung als Verkaufsargument bei neuen Maschinen eingesetzt werden kann; *d*) die Priorisierung bei vorhandenen Installationen implementiert werden kann und die Kunden dadurch einen Mehrwert erhalten.

4.2. Erweiterbarkeit

Die von der axxeo GmbH angebotenen Maschinen sind auf einen Ausfallzeitraum von 3-5 Minuten ausgelegt. Das Web-GUI könnte um eine Funktion erweitert werden, die ermöglicht, das S-A-R zu aktivieren bzw. auf statisches Routing umzustellen. Eine andere Möglichkeit wäre die Erstellung eines Monitoring-Scripts, welches die physikalische Verbindung, die Funktionen der NIC¹ und des Routers davor überwacht und entsprechend das S-A-R aktiviert oder deaktiviert.

¹Network Interface - Netzwerkkarte

Die Traffic-Priorisierung lässt sich durch eine Anpassung der tcng(1)-Scripte erweitern; mithilfe von weiteren Tests könnten neben HTB auch andere Algorithmen verwendet werden. Ein weiterer Punkt ist die Priorisierung von einzelnen HTB-leaf-Klassen, um beispielsweise bei der Nutzung von VoIP² Bandbreite bereitzustellen (hohe Priorität), aber bei geringer Nutzung allen anderen Klassen (geringe Priorität) Bandbreite zu gewähren (Brown, 2006a, Kapitel 7.1.4).

4.3. Alternativen

Eine Alternative wäre die Juniper SSG20 (Juniper Networks, 2009), welche ca. 850€ (incl. MwSt.) kosten würde (TLK Distributions, 2009). Hier fehlen allerdings noch Folgekosten für *a)* Schulungen, da die Software nicht bekannt ist, und für *b)* Hardware-Garantie/Hardware-Austausch, da die Maschinen spezielle Hardware verwenden. Außerdem wäre der Einsatz einer solchen Maschine ein Bruch der Windows-/Linux-Homogenität in vielen Kunden-Netzen. Die Linux-Administratoren der Kunden müssten bei Problemen mit der axxeo GmbH erst die Syntax und Semantik der Juniper erlernen, anstatt auf einem bereits bekannten System zu arbeiten.

Tätigkeit	geplante Dauer	benötigte Dauer	Unterschied
Basiskonzept, Information, etc.	3-4	10	+6
Installation, Konfiguration, Einbinden in bestehende Scripte	5	5	0
Anpassung von Parametern in Absprache mit Auftraggeber	1	1	0
Test des Aufbaus, Zusammenspiel von vorhandenen Systemscripten mit neuer Konfiguration bei Reboot, Anpassung etc.	4	6	+2
Erstellung von Scripten, um Leitungs-Ausfall festzustellen	3	0	-3
Laufzeit-Tests (Last-Verteilung, Priorisierung), Simulation von versch. Ausfällen (Netzwerk-Interface, Next-Hop-Router, ...)	7	3	-4
Dokumentation	8	10	+2
variable Pufferzeit	3	0	-3
Ergebnis	35	35	0

Tabelle 4.2.: Gegenüberstellung der geplanten und benötigten Zeit in h

²Voice over Internet Protocol, Internet-Telefonie

A. Quellen

- [Balliache 2003] BALLIACHE, Leonardo: *Differentiated Service on Linux HOWTO*. Webseite, abgerufen am 18.10.2009. 2003. – <http://www.opalsoft.net/qos/DS.htm>
- [Brown 2006a] BROWN, Martin A.: *Traffic Control HOWTO*. Webseite, abgerufen am 18.10.2009. 2006. – <http://tldp.org/HOWTO/Traffic-Control-HOWTO/index.html>
- [Brown 2006b] BROWN, Martin A.: *Traffic Control using tcng and HTB HOWTO*. Webseite, abgerufen am 18.10.2009. 2006. – <http://tldp.org/HOWTO/Traffic-Control-tcng-HTB-HOWTO/index.html>
- [Coene 2005] COENE, Stef: *Docum - HTB*. Webseite, abgerufen am 18.10.2009. 2005. – <http://www.docum.org/docum.org/docs/>
- [Hubert u. a. 2005] HUBERT, Bert ; GRAF, Thomas ; MAXWELL, Greg ; MOOK, Remco van ; OOSTERHOUT, Martijn van ; SCHROEDER, Paul B. ; LARROY, Pedro: *Linux Advanced Routing & Traffic Control*. Webseite, abgerufen am 18.10.2009. 2005. – <http://lartc.org/howto/>
- [Juniper Networks 2009] JUNIPER NETWORKS, Inc.: *SSG20 Secure Services Gateway - Small Office Network Security Platform*. Webseite, abgerufen am 18.10.2009. 2009. – <http://www.juniper.net/us/en/products-services/security/ssg-series/ssg20/>
- [TLK Distributions 2009] TLK DISTRIBUTIONS, GmbH: *Juniper NetScreen Preisübersicht*. Webseite, abgerufen am 18.10.2009. 2009. – http://gast:gast@tlk.de/cgi/listen.cgi?ZYJUN_SSG5

B. Glossar

bucket engl. für Eimer. 6

Cache Zwischenspeicher. 9

Debian Linux-Distribution. 1

egress engl. für Ausgehend. 5

hping Tool um Netzwerkpakete zu erzeugen. 9

ingress engl. für Eingehend. 5

Policy engl. für Richtlinie. 4

qdisc Scheduler (regelt die zeitliche Ausführung mehrerer Prozesse) – der Standard-Scheduler für Linux-Netzwerkinterfaces ist FIFO. 5

Routing Funktion um Wege bei der Nachrichtenübermittlung festzulegen. 4

tc(8) Tool zur Kontrolle des Netzwerkverkehrs im Linux-Kernel. 1, 4, 5, 8

tcng(1) Tool zur Kompilierung von Scripten in tc-Befehle. 4, 5, 8, 11

token engl. für Jeton, Wertmarke, o. ä. . 6

C. Akronyme

HTB Hierarchical Token Bucket packet scheduler – eine Variante des *Token-Bucket-Filter*.
5, 11

HTTP Hypertext Transfer Protocol, Netzwerkprotokoll zur Übertragung von Webseiten. 1

IP Internet Protocol. 3, 8

ISP Internet Service Provider / Internetdienstanbieter. 1

NIC Network Interface - Netzwerkkarte. 10

S-A-R Split-Access-Routing, Internetzugang über mehrere Anbindungen gleichzeitig. 4, 8,
10

SDSL Symmetric Digital Subscriber Line. 5

SMTP Simple Mail Transfer Protocol, Netzwerkprotokoll zur Übertragung von E-Mails. 1

SSH Secure Shell, sicherer Ersatz für *telnet*. 1

VDSL2+ Extended bandwidth Asymmetric Digital Subscriber Line 2. 5

VoIP Voice over Internet Protocol, Internet-Telefonie. 11

D. Abbildungsverzeichnis

2.1.	Netzplan	3
2.2.	gewünschte Bandbreite	6
2.3.	Bandbreite nach HTB	6
2.4.	grafische Darstellung des TCC-Scripts von eth1 (siehe E.7 auf Seite 19)	7
E.1.	Bestätigung über die durchgeführte Projektarbeit	21

E. Verzeichnis der Listings

E.1.	Routing vor Veränderung	16
E.2.	Routing nach Veränderung	16
E.3.	Laufende Konfiguration der Traffic-Kontrolle	16
E.4.	Bei Firewall-start eingelesenes Script - erweitert um Split-Routing-Setup . .	17
E.5.	Init-Sript für Priorisierung	18
E.6.	Konfigurationsdatei für Routing	19
E.7.	tcng(1)-Script zur Erstellung von tc(8)-Befehlen für eth1	19
E.8.	tcng(1)-Script zur Erstellung von tc(8)-Befehlen für eth2	20

Listing E.1: Routing vor Veränderung

```

1 root@firewall:/# ip r
2 82.100.226.144/28 dev eth0 proto kernel scope link src 82.100.226.145
3 10.1.0.0/24 dev eth1 proto kernel scope link src 10.1.0.1
4 default via 10.1.0.2 dev eth1

```

Listing E.2: Routing nach Veränderung

```

1 root@firewall:/# ip r
2 82.100.226.144/28 dev eth0 proto kernel scope link src 82.100.226.145
3 10.2.0.0/24 dev eth2 proto kernel scope link src 10.2.0.1
4 10.200.0.0/24 via 10.2.0.1 dev eth2
5 10.1.0.0/24 dev eth1 proto kernel scope link src 10.1.0.1
6 10.100.0.0/24 via 10.1.0.1 dev eth1
7 default
8     nexthop via 10.1.0.2 dev eth1 weight 1
9     nexthop via 10.2.0.2 dev eth2 weight 10

```

Listing E.3: Laufende Konfiguration der Traffic-Kontrolle

```

1 root@firewall:~$ tc -d class ls dev eth1
2 class htb 2:1 root rate 200000bit ceil 200000bit burst 24999b/8 mpu 0b overhead 0b cburst 24999b/8 mpu 0b overhead 0b level 7
3 class htb 2:2 parent 2:1 leaf 3: prio 0 quantum 1000 rate 50000bit ceil 100000bit burst 5Kb/8 mpu 0b overhead 0b cburst 12799b/8 mpu 0b overhead 0b level 0
4 class htb 2:3 parent 2:1 leaf 4: prio 0 quantum 1000 rate 50000bit ceil 100000bit burst 6399b/8 mpu 0b overhead 0b cburst 12799b/8 mpu 0b overhead 0b level 0
5 class htb 2:4 parent 2:1 leaf 5: prio 0 quantum 1250 rate 100000bit ceil 140000bit burst 12799b/8 mpu 0b overhead 0b cburst 17919b/8 mpu 0b overhead 0b level 0
6 root@firewall:~$ tc -d qdisc ls dev eth1
7 qdisc dsmark 1: root indices 0x0004 default_index 0x0000
8 qdisc htb 2: parent 1: r2q 10 default 0 direct_packets_stat 0 ver 3.17
9 qdisc sqf 3: parent 2:2 limit 127p quantum 1514b flows 127/1024
10 qdisc sqf 4: parent 2:3 limit 127p quantum 1514b flows 127/1024
11 qdisc sqf 5: parent 2:4 limit 127p quantum 1514b flows 127/1024

```

```

12 root@firewall:~$ tc -d class ls dev eth2
13 class htb 2:1 root rate 100000Kbit ceil 100000Kbit burst 12800Kb/8 mpu 0b overhead 0b cburst 12800Kb/8 mpu 0b
   overhead 0b level 7
14 class htb 2:2 parent 2:1 leaf 3: prio 0 quantum 1600 rate 128000bit ceil 1000Kbit burst 16Kb/8 mpu 0b overhead 0b
   cburst 128Kb/8 mpu 0b overhead 0b level 0
15 class htb 2:3 parent 2:1 leaf 4: prio 0 quantum 1600 rate 128000bit ceil 1000Kbit burst 15743b/8 mpu 0b overhead 0b
   cburst 128Kb/8 mpu 0b overhead 0b level 0
16 class htb 2:4 parent 2:1 leaf 5: prio 0 quantum 12500 rate 1000Kbit ceil 99000Kbit burst 128Kb/8 mpu 0b overhead 0b
   cburst 12672Kb/8 mpu 0b overhead 0b level 0
17 root@firewall:~$ tc -d qdisc ls dev eth2
18 qdisc dsmark 1: root indices 0x0004 default_index 0x0000
19 qdisc htb 2: parent 1: r2q 10 default 0 direct_packets_stat 0 ver 3.17
20 qdisc sqf 3: parent 2:2 limit 127p quantum 1514b flows 127/1024
21 qdisc sqf 4: parent 2:3 limit 127p quantum 1514b flows 127/1024
22 qdisc sqf 5: parent 2:4 limit 127p quantum 1514b flows 127/1024

```

Listing E.4: Bei Firewall-start eingelesenes Script - erweitert um Split-Routing-Setup

```

1 # extra routes
2 # Eintraege vor der FOXCONFIG Zeile sind
3 # fuer die Oberflaeche unsichtbar
4
5 # __split-access-routing__
6
7 # get variables
8 source /usr/local/etc/tc-routing.sh
9
10 existing_routes_to_table ()
11 {
12     $IP route list table main | grep -vE "default|nexthop" | \
13         while read line; do
14             $IP r r $line table $2
15         done
16 }
17
18 default_rules_routes ()
19 {
20     $IP r r 127.0.0.0/8 dev lo table $2
21     $IP r r default dev $1 table $2
22     $IP r flush cache
23 }
24
25 table=$T1
26 existing_routes_to_table $EDEV1 $table
27 default_rules_routes $EDEV1 $table
28 $IP rule list | grep -q "from $EDEV1_IP lookup $table" || $IP rule add from $EDEV1_IP table $table
29
30 table=$T2
31 existing_routes_to_table $EDEV2 $table
32 default_rules_routes $EDEV2 $table
33 $IP rule list | grep -q "from $EDEV2_IP lookup $table" || $IP rule add from $EDEV2_IP table $table
34
35 # default route (set the actual split)
36 $IP r r default scope global nexthop via $EDEV1_R dev $EDEV1 weight $EDEV1_WEIGHT nexthop via $EDEV2_R dev
   $EDEV2 weight $EDEV2_WEIGHT
37
38 # set fixed routes for eth1
39 while read line; do

```

```

40     $IP r r $line via $EDEV1_IP
41 done < /usr/local/etc/fixed-routes-eth1
42
43 # set fixed routes for eth2
44 while read line; do
45     $IP r r $line via $EDEV2_IP
46 done < /usr/local/etc/fixed-routes-eth2

```

Listing E.5: Init-Skript für Priorisierung

```

1  #!/bin/sh
2
3 source /usr/local/etc/tc-routing.sh
4
5 clear_qdiscs ()
6 {
7     # clear up- and downlink qdiscs
8     for device in $DEVLIST ; do
9         $TC qdisc del dev $device ingress 2>&1 > /dev/null
10        $TC qdisc del dev $device root 2>&1 > /dev/null
11    done
12 }
13
14 create_qdiscs ()
15 {
16     $TCNG /usr/local/etc/tc/eth1.tcc | sed s,`tc,$TC, | $BASH
17     $TCNG /usr/local/etc/tc/eth2.tcc | sed s,`tc,$TC, | $BASH
18 }
19
20
21 case "$1" in
22     status)
23         for device in $EDEVLIST ; do
24             $TC -s qdisc ls dev $device
25             $TC -s class ls dev $device
26         done
27         ;;
28     start)
29         clear_qdiscs ;
30         create_qdiscs ;
31         ;;
32     stop)
33         clear_qdiscs ;
34         ;;
35     restart )
36         clear_qdiscs ;
37         create_qdiscs ;
38         ;;
39     *)
40         echo "Usage: $0 {start|stop|restart|status}"
41         exit 1
42         ;;
43 esac
44
45 exit 0
46
47 # vim:tabstop=2:expandtab:shiftwidth=2

```

Listing E.6: Konfigurationsdatei für Routing

```

1 #! /bin/sh
2 # generic config used in /usr/local/etc/network/if-up.d/split-access-routing and /usr/local/bin/traffic-shaper.sh
3
4 # used programs
5 BASH=/bin/bash
6 IP=/bin/ip
7 IPT=/sbin/iptables
8 TC=/sbin/tc
9 TCNG=/usr/bin/tcng
10
11 # external Interfaces
12 EDEV1=eth1
13 EDEV2=eth2
14 EDEVLIST="eth1 eth2"
15
16 # routing-tables
17 T1=one
18 T2=two
19
20 # IPs
21 EDEV1_IP=10.1.0.1
22 EDEV2_IP=10.2.0.1
23
24 # next-hop-router
25 EDEV1_R=10.1.0.2
26 EDEV2_R=10.2.0.2
27
28 # balancing
29 EDEV1_WEIGHT=1
30 EDEV2_WEIGHT=1

```

Listing E.7: tcng(1)-Script zur Erstellung von tc(8)-Befehlen für eth1

```

1 dev "eth1"{
2     egress {
3         class ( <$interactive_out> )           // Services we access
4             if ip_tos_delay == 1 && tcp_dport == 22 // SSH + desire for short delay,
5
6
7             if ip_len < 256 && tcp_dport == 22 // outdated by RFC2474, but some programs might still use
8                 if tcp_dport == 53 || udp_dport == 53 // keep ip_tos* at the front of the expression,
9                     if ip_len < 512 && tcp_dport == 80; // is early ip-header-field
10
11
12         class ( <$interactive_in> )           // short SSH-packets
13             if ip_tos_delay == 1 && tcp_sport == 22 // DNS
14
15             if ip_len < 256 && tcp_sport == 22 // short HTTP
16
17         class ( <$other> )      if 1;           // Answers on existant connections (e.g. ACKs)
18
19
20     // Configure how much bandwidth the classes get
21     htb () {                                // Hierarchical Token Bucket packet scheduler
22
23

```

```

24     class ( rate 200kbps, ceil 200kbps, burst 200000b, cburst 200000b ) {           // rate = how much is reserved
25         , ceil = how much it can use max
26         // Allow $interactive to use up to 100kbps, but make sure there's always 50kbps available
27         $interactive_out = class ( rate 50kbps, ceil 100kbps, burst 50000b, cburst 100000b ) { sfq; } ; // Stochastic Fairness Queueing,
28                                         // inhibits dominations of one connection inside the class
29         $interactive_in = class ( rate 50kbps, ceil 100kbps, burst 50000b, cburst 100000b ) { sfq; } ;
30         $other = class ( rate 100kbps, ceil 140kbps, burst 100000b, cburst 140000b ) { sfq; } ;
31     }
32 }
33 }
```

Listing E.8: tcng(1)-Script zur Erstellung von tc(8)-Befehlen für eth2

```

1 dev "eth2"{
2     egress {
3         class ( <$interactive_out> )                                // Services we access
4             if ip_tos_delay == 1 && tcp_dport == 22                // SSH + desire for short delay,
5                                         // outdated by RFC2474, but some programs might still use
6                                         // keep ip_tos* at the front of the expression,
7                                         // is early ip-header-field
8             if ip_len < 256 && tcp_dport == 22                      // short SSH-packets
9             if tcp_dport == 53 || udp_dport == 53                     // DNS
10            if ip_len < 512 && tcp_dport == 80;                      // short HTTP
11
12         class ( <$interactive_in> )                                // Answers on existant connections (e.g. ACKs)
13             if ip_tos_delay == 1 && tcp_sport == 22                // SSH + desire for short delay,
14                                         // outdated by RFC2474, but some programs might still use
15             if ip_len < 256 && tcp_sport == 22                      // short SSH-packets
16
17         class ( <$other> )      if 1;                                // Always returns true. Leftover packets
18
19
20     // Configure how much bandwidth the classes get
21     htb () {                                                 // Hierarchical Token Bucket packet scheduler
22                                         // <http://luxik.cdi.cz/~devik/qos/htb/>
23
24         class ( rate 50Mbps, ceil 50Mbps ) {                  // rate = how much is reserved, ceil = how much it can use
25             max
26             // Allow $interactive to use up to 1Mbps, but make sure there's always 128kbps available
27             $interactive_out = class ( rate 128kbps, ceil 1Mbps, burst 128kb, cburst 1Mb ) { sfq; } ; // Stochastic
28                                         // Fairness Queueing,
29                                         // inhibits dominations of one connection inside the class
30             $interactive_in = class ( rate 128kbps, ceil 1Mbps, burst 123kb, cburst 1Mb ) { sfq; } ;
31             $other = class ( rate 1Mbps, ceil 49Mbps, burst 1Mb, cburst 49Mb ) { sfq; } ;
32         }
33     }
```

 Industrie- und Handelskammer Hannover		
Bestätigung über die durchgeführte betriebliche Projektarbeit		
Antragsteller:	Ausbildungsbetrieb (Praktikumsbetrieb):	
Name: Lorenzen	Axxeo GmbH	
Vorname: Oluf		
Straße: Limmerstraße 49	Kniestraße 27	
PLZ, Ort: 30451	30167	
Email: ol@axxeo.de	ag@axxeo.de	
Projektbezeichnung (Auftrag/Teilauftrag): Split-Access-Routing mit Priorisierung auf Linux-Basis.		
12.10.09 Projektbeginn	30.10.09 Projektende	35 Zeitaufwand in Stunden
Bestätigung des Ausbildungsbetriebes/Praktikumsbetriebes: Wir versichern, dass das Projekt wie in der Dokumentation dargestellt, in unserem Unternehmen realisiert wurde.		
Godzina, Andreas Name, Vorname (Projektverantwortlicher)	A. Gahr Stempel u. Unterschrift (Ausbildender)	
Verbindliche Erklärung des Prüfungsteilnehmers/der Prüfungsteilnehmerin: Ich versichere, dass ich das Projekt und die dazugehörige Dokumentation selbstständig erarbeitet habe.		
Hannover, 23.11.09 Ort, Datum	Olaf Gahr Unterschrift des Prüflings	
Diese Bestätigung ist in Druckschrift oder maschinell auszufüllen und als letzte Seite in die Projektdokumentation einzufügen.		

1/1

Abbildung E.1.: Bestätigung über die durchgeführte Projektarbeit