INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Diplomarbeit in Informatik

Konzeption und prototypische Umsetzung eines WAN-Gateway zur inter-organisationalen, dienstgütespezifischen Pfadschaltung

Daniel Feuchtinger

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Diplomarbeit in Informatik

Konzeption und prototypische Umsetzung eines WAN-Gateway zur inter-organisationalen, dienstgütespezifischen Pfadschaltung

Daniel Feuchtinger

Aufgabensteller: Prof. Dr. Dieter Kranzlmüller

Betreuer: Dr. Vitalian Danciu
Dipl.-Inf. Martin Metzker

Abgabetermin: 28. Dezember 2011

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 28. Dezember 2011

.....

(Unterschrift des Kandidaten)

Kurzfassung

Streaming Media, VoIP und Online-Spiele sind nur einige Beispiele von Anwendungen, die spezifische Qualitätsanforderungen an Verbindungen über das Internet stellen. Quality-of-Service (QoS) ist bisher weitgehend durch die Grenzen autonomer Systeme (AS) beschränkt. In der Regel fehlt die Möglichkeit, einen Pfad zu finden, der eine Dienstgüte garantiert, weiterhin ist nicht gesichert, dass dieser Pfad auch tatsächlich verwendet wird, da das Routing in anderen autonomen Systemen nicht kontrolliert werden kann. Die Akzeptanz einer Lösung dieser Probleme hängt wesentlich davon ab, dass sie attraktiv für die Netzbetreiber und mit geringem Aufwand realisierbar ist.

Diese Arbeit macht sich genau das zur Aufgabe: Auf IP-Ebene soll ein Pfad mit einer spezifische Dienstgüte von Ende zu Ende geschaltet werden, ohne große Anschaffungen und Umstellungen bei den Netzbetreibern zu erfordern. Das wird erreicht, indem jedes teilnehmende AS mindestens ein Gateway zur Verfügung stellt (oder einen Router um die Funktionalität der Gateways erweitert), das Verbindungsanfragen entgegennimmt und einen Pfad durch die autonomen Systeme entlang der Gateways sucht. Entlang dieses Pfades werden Ressourcen reserviert und die IP-Pakete zur Einhaltung des Pfades von Gateway zu Gateway bzw. von AS zu AS weitergeleitet. Die Reservierung von Ressourcen und das Routing entlang des Pfades werden ohne Eingriffe in fremde autonome Systeme und ohne notwendige Anpassungen des Routing realisiert.

Abstract

Streaming media, VoIP and online games are some examples of applications that have specific quality of service (QoS) requirements on network connections over the Internet. So far, QoS is mostly limited by the borders of autonomous systems. Generally, it is impossible to find a path with a certain quality of service, and if so, it is by no means guaranteed that this path actually is being used, since there is no way to control the routing in other autonomous systems. If a solution to this problem is accepted sustantially depends on whether or not it is attractive to internet service providers and whether or not it can be implemented at low costs.

The present diploma thesis focuses on this point: How to negotiate a path with a certain quality of service from end to end without requiring a lot of investments and changes by service providers. This is achieved through a gateway in every participating AS which accepts connection requests and searches a path through the autonomous systems following the gateways. Along this path resources are allocated. The IP packets are forwarded from gateway to gateway in order to maintain the path. Resource allocation and routing along the path is achieved without intervening in autonomous systems and without any modification to the routing concept being necessary.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Klassische Technologien wie Radio, Fernsehen, Printmedien und Telefon bekommen immer mehr Konkurrenz durch neue Dienste im Internet. Während die ursprünglichen Aufgaben des Internets, wie die Übertragung von HTML-Seiten oder E-Mails, kaum Anforderungen an die Dienstgüte (Quality of Service / QoS) der IP-Übertragung stellen, setzen die neuen Dienste eine spezifische Übertragungsqualität voraus. Ein Telefongespräch über das Internet benötigt (unter anderem) eine konstante Übertragungsrate von ca. 100 KBit/s, um eine gute Sprachqualität zu gewährleisten (ISDN benötigt 64 KBit/s, dazu kommt noch, in Abhängigkeit von den Paketgrößen, Overhead durch Header der Schichten TCP, IP und Schicht 2). Es konkurrieren Dienste mit sehr unterschiedlichen Anforderungen um Ressourcen in einem Netz, das ursprünglich jeden Datenverkehr gleich behandeln sollte.

Prinzipiell ist die passende Behandlung des Datenverkehrs verschiedener Dienste in IP-Netzen inzwischen zwar möglich, dabei ergibt sich aber folgendes Problem: Da das Internet ein Verbund vieler selbstständig organisierter Subnetze (*autonomer Systeme*) ist, liegt die Qualität und Vermittlung des Datenverkehrs einer Verbindung, die mehrere autonome Systeme passiert, in den Händen verschiedener Organisationen, die oft unterschiedliche Technologien einsetzen und unterschiedliche Interessen haben. Die Pakete einer Ende-zu-Ende-Verbindung passieren also Vertrauensgrenzen (Trust-Boundaries), Technologiegrenzen, sowie Zuständigkeits- und Interessensgrenzen, die einerseits zwischen den autonomen Systemen, andererseits zwischen Netzbetreiber (Internet-Service-Provider bzw. ISP) und Endnutzer verlaufen können (siehe Abschnitt 1.1). An diesen Grenzen muss es eine Möglichkeit geben, der Berechtigung zur Bevorzugung zu vertrauen (beim Passieren der Trust-Boundary), die Bevorzugung in die jeweils verfügbare Technik zu übersetzen (beim Passieren der Technologiegrenzen) und sich den mit der Bevorzugung verbundenen Aufwand mit einer Gegenleistung vergelten zu lassen (an der Zuständigkeits- und Interessensgrenze). Die Bereitstellung dieser Möglichkeiten ist, nicht zuletzt wegen der Reservierung von Ressourcen, mit einigem Aufwand verbunden. Daher ist es sinnvoll, sie auf einen vorher geschalteten Pfad zu beschränken.

In Abbildung 1.1 sind diese Probleme an einem Beispiel dargestellt. AS1 und AS5 setzen für QoS im eigenen Netz IntServ und RSVP ([BCS 94] und [BZB+ 97]) ein, für den Datenverkehr über AS2 stehen IntServ und RSVP aber nicht zur Verfügung (auch nicht RSVP-TE [ABG+ 01]). Sowohl die Pfadschaltung und Ressourcenreservierung (RSVP) als auch die Zuteilung der Ressourcen zu einer Verbindung (IntServ) enden an den Grenzen von AS1 und AS5. Wegewahl und QoS in den autonomen Systemen dazwischen werden mit MPLS ([RVC 01]) realisiert. Die QoS-Parameter, die von AS1 und AS5 für die Verbindung vorgesehen sind, müssen in Parameter für IntServ/RSVP und MPLS übersetzt werden. Außerdem muss am Übergang von AS2 zu AS3 und AS4 feststehen, welchen Weg die Pakete

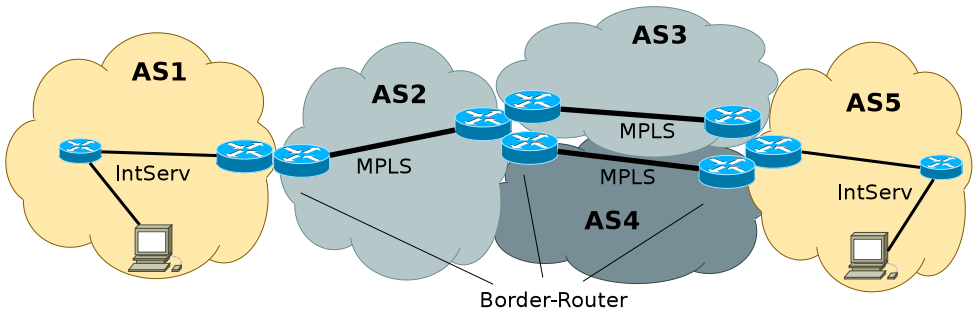


Abbildung 1.1.: Heterogenität als Problem bei der QoS-Pfadschaltung über die Grenzen autonomer Systeme hinweg

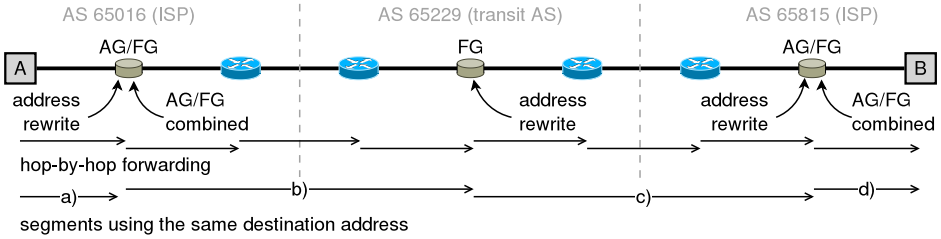


Abbildung 1.2.: Pfadschaltung in [DKMY 11]

nehmen, um eine doppelte Reservierung von Ressourcen zu vermeiden.

In [DKMY 11] wird vorgeschlagen, den Pfad für eine Ende-zu-Ende-Verbindung durch Standard-IP-Routing, kombiniert mit speziellen, von den Netzbetreibern dafür reservierten IP-Adressen, zu realisieren. Den Endpunkten wird von *Access-Gateways* (AGs) eine IP-Adresse mitgeteilt, die einen Kommunikationskanal zum anderen Ende repräsentiert und für die Kommunikation über diesen Kanal verwendet werden kann. In Abbildung 1.2 bekommt Ende B aus Sicht von Ende A eine zusätzliche IP-Adresse, über die QoS-gesteuerte Kommunikation möglich ist. Diese zusätzliche IP-Adresse stammt aus einem für diesen Zweck vorgesehenen Subnetz des AG-Betreibers auf der Seite von A. Zwischen den Access-Gateways wird der Pfad durch weitere IP-Adressen festgelegt, die je aus einem Subnetz der Betreiber des *Forwarding-Gateways* (FG) stammen. Die FGs adressieren Pakete, die an die Adresse aus dem eigenen Subnetz adressiert sind, an die entsprechende Adresse aus dem Subnetz des nächsten FGs auf dem Pfad (`address rewrite` in Abbildung 1.2). Diese Idee ist die Grundlage der Gateways für die dienstgütespezifische Pfadschaltung, die in dieser Diplomarbeit entwickelt werden.

1.1. Terminologie

In diesem Abschnitt wird vorab die Verwendung einiger Begriffe erläutert.

Quality of Service (QoS) bezieht sich, soweit nicht anders angegeben, auf Eigenschaften (Qualitäten) der Übertragung von IP-Paketen (Service). Ein QoS-Parameter ist eine bestimmte Eigenschaft dieser Übertragung. Die Arbeit orientiert sich zwar im Wesentlichen an vier verbreiteten QoS-Parametern (*Übertragungsrate*, *Latenz*, *Jitter* und *Paketverlustrate*), die Beschaffenheit eines QoS-Parameters wird dadurch aber nicht eingeschränkt. In [Tane 02] ist statt der Paketverlustrate als grundlegender Parameter die Zuverlässigkeit der Übertragung definiert. Im Internet wird heute in der Regel auf Schicht 2 mittels einer Prüfsumme die korrekte Übertragung überprüft und bei einem Fehler das Paket verworfen; das macht die Zuverlässigkeit als Parameter weniger interessant. Paketverluste bzw. genauer Packet-Drops sind zudem ein Mittel zur Stau- und Fluss-Steuerung und für QoS interessant (Packet-Drops kann man steuern, Verfälschungen nur erkennen), daher wird in dieser Arbeit als vierter Parameter die Paketverlustrate gewählt. Prinzipiell ist ein QoS-Parameter eine Eigenschaft der IP-Übertragung, die der Diensterbringer (z. B. der Internet-Service-Provider) dem Dienstnutzer (z. B. einem Kunden) zur Verfügung stellen kann.

Weitere Beispiele für spezifische QoS-Parameter sind *Verfügbarkeit*, *Priorität*, *Erhaltung der Paketreihenfolge*, *Abhörsicherheit* (Übertragungsmedien werden überwacht, Daten werden auf unsicheren Abschnitten verschlüsselt) und *Zuverlässigkeit* (die gesendeten Daten kommen unverfälscht an). Die Parameter können (und müssen, z. B. im Fall von „Priorität“) weiter differenziert werden. Beispielsweise kann für die Latenz ein Maximal- und ein Durchschnittswert angegeben werden, für die Übertragungsrate kann die Regelmäßigkeit festgelegt werden (können bei einer Rate von 50 KBit/s alle 50 KBit in einer Millisekunde eintreffen, oder ist ein gleichmäßigerer Datenfluss garantiert?), und die Priorität kann sich darauf beziehen, wer bei einem Leitungsausfall die nächste freie Leitung bekommt. Durch die Differenzierung werden die jeweiligen Parameter an die Bedürfnisse der Anwendungen angepasst: Die Videoübertragung eines Films hat, dank Pufferung, andere Anforderungen an die Gleichmäßigkeit des Datenflusses als die Videokonferenz, auch wenn die Übertragungsrate im

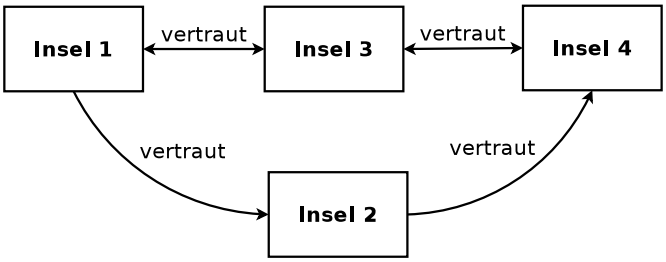


Abbildung 1.3.: Vier Vertrauensinseln sind untereinander durch *Chains-of-Trust* verknüpft

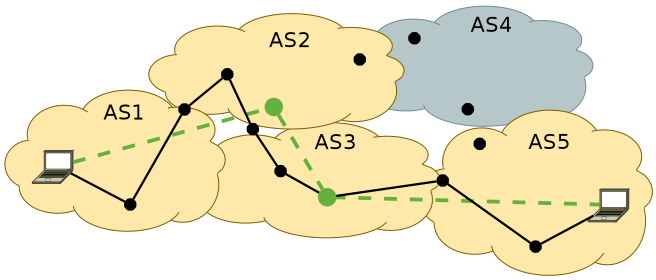


Abbildung 1.4.: Ein *Pfad* durch die autonomen Systeme: Die IP-Hops bzw. Router sind mit durchgezogenen Linien verbunden, den davon abstrahierten Pfad markiert die gestrichelte Linie

Schnitt bei beiden gleich sein kann.

Ein *autonomes System* (AS) ist ein zusammenhängender Teilbereich des Internets, der selbstständig organisiert ist und dessen Betreiber insbesondere die Kontrolle über das Routing und die Ressourcen innerhalb dieses Teilbereichs haben und darüber entscheiden können, ob neue Ressourcen (Hardware und Software) eingesetzt werden. Diese Definition stimmt nicht ganz mit den Definitionen in RFC 1771 und RFC 1930 ([ReLi 95] und [HaBa 96]) überein, da ein autonomes System sich dort z. B. auch über eine eigene ASN (AS-Number) definiert. Die Verwendung des Begriffs in dieser Arbeit ist also abstrakter, autonome Systeme nach den genannten RFCs fallen aber auch unter diese Definition. Die Betreiber eines autonomen Systems werden hier auch *Netzbetreiber* genannt.

Eine Vertrauensgrenze (*Trust-Boundary*) umschließt in dieser Arbeit einen (topologischen) Teil eines Netzes (Vertrauensinsel), in dem Nachrichten oder Entitäten ein bestimmtes Vertrauenslevel besitzen. Folgendes Beispiel verdeutlicht den Begriff: Ein Router innerhalb eines autonomen Systems vertraut darauf, dass die IP-Nachrichten, die von einem anderen Router (Entität) aus demselben AS kommen, Senderadressen besitzen, die nicht mehr überprüft werden müssen (abgesehen davon, dass das gar nicht mehr möglich wäre), d. h. diese Nachrichten (bzw. der Router) besitzen das Vertrauenslevel „spoofing-geprüft“. Die Vertrauensgrenze verläuft in diesem Beispiel zwischen Endkunden und ISP: An den Netzzugangspunkten kann nicht darauf vertraut werden, dass die Kunden nur die ihnen zugewiesene IP-Adresse als Absender verwenden, hier ist Kontrolle nötig. Eine *Chain-of-Trust* bezeichnet einen Zusammenschluss mehrerer Vertrauensinseln, wobei sich das Vertrauen über einen Pfad transitiv fortsetzt (Abbildung 1.3: Insel 1 vertraut Insel 2 und Insel 2 vertraut Insel 4, daher vertraut Insel 1 implizit auch Insel 4; auf der Abbildung existieren Chains-of-Trust zwischen allen Inseln, d. h. jeder vertraut jedem). Übertragen auf das Spoofing-Beispiel bedeutet das, dass ein ISP darauf vertrauen kann, dass der Nachbar-ISP an seinen Vertrauensgrenzen Spoofing verhindert oder wiederum seinen Nachbarn vertrauen kann und daher der ankommende Datenverkehr ebenfalls das Vertrauenslevel „spoofing-geprüft“ erhält.

Ein *Gateway* bezeichnet in dieser Arbeit immer eine funktionale Einheit im Netz, das in dem autonomen System, in dem es sich befindet, für die Schaltung von Pfaden mit spezifischen QoS-Parametern zuständig ist. Ein *Pfad* bezeichnet die Menge von autonomen Systemen, die ein IP-Paket von einer Quelle bis zu einer Senke durchläuft, wobei ein Knoten ein Gateway, die Quelle oder Senke bzw.

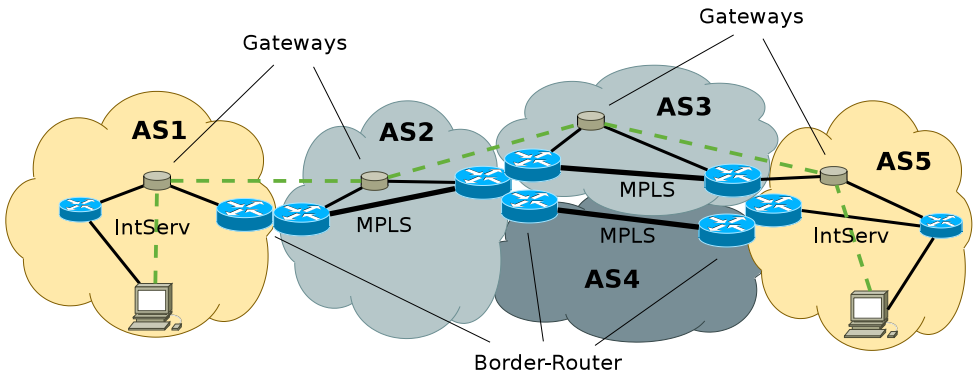


Abbildung 1.5.: Gateways schalten einen Pfad (gestrichelte Verbindungslinien) durch die autonomen Systeme

das je zugehörige autonome System repräsentiert (siehe Abbildung 1.4). Das *nächste bzw. vorherige Gateway* bezeichnet das aus Sicht des aktuellen Gateways jeweils vorherige bzw. nächste Gateway bezüglich eines (potentiellen) Pfades. Die Pakete einer Verbindung müssen ein Gateway nicht passieren, es ist lediglich für die Pfadschaltung im jeweiligen AS zuständig. Ein Gateway kann aber auch gleichzeitig andere Aufgaben erfüllen, z. B. Routing. In diesem Fall könnten die Pakete das Gateway passieren. *Über ein Gateway routen* bedeutet entsprechend nicht unbedingt, dass die Pakete dieses Gateway tasächlich passieren, sondern dass die Pakete über Router im Zuständigkeitsbereich dieses Gateways geleitet werden.

Der Begriff *lokal* wird im Sinne von „im Zuständigkeitsbereich dieses Gateways“, d.h. „innerhalb dieses autonomen Systems“, verwendet, *global* wird entsprechend mit „AS-übergreifend“ gleichgesetzt.

Eine *Verbindung* ist ein Pfad mit dazugehörigen QoS-Parametern, der von den Enden genutzt werden kann. Es wird zwischen *Verbindungsauslösung* und *Verbindungsaufbau* unterschieden. Die Verbindungsauslösung bezeichnet die (autorisierte) Übergabe eines Verbindungswunsches an einen Verbindungsdienst, der Verbindungsaufbau bezeichnet die Pfadsuche, Pfadschaltung und Ressourcenreservierung, die der Verbindungsdienst ausführt.

Address-Rewriting bezeichnet das Ändern der Empfänger- und, je nach Anforderung, der Absenderadresse in allen Datenpaketen, die zu einer Verbindung gehören (`address rewrite` in Abbildung 1.2). Handelt es sich um eine unidirektionale Verbindung, so wird nur die Empfängeradresse geändert, bei einer bidirektionalen Verbindung auch die Absenderadresse. Die neue Empfängeradresse, die für die Vermittlung zum nächsten Gateway dient, wird die *Rewriting-Adresse* genannt.

1.2. Aufgabenstellung

Ziel dieser Arbeit ist ein Konzept für die Schaltung dienstgütespezifischer Kommunikationskanäle im Internet über AS-Grenzen hinweg. Dafür ist ein Gateway konzipiert, das die Pfadschaltung für die Kommunikationskanäle durchführen kann (Abbildung 1.5). Das Gateway ist für QoS des Pfades im eigenen AS verantwortlich und muss dafür sorgen, dass die zu dem Kommunikationskanal gehörenden Datenpakete in das nächste AS auf dem Pfad vermittelt werden.

Die Funktionalität der Gateways muss für die Konzeption in Funktionseinheiten aufgeteilt werden, die wiederum auf Rollen und Komponenten abgebildet werden können. Daraus muss ein *Rollenmodell* erstellt und die Verteilung von Rollen auf neue und bestehende Komponenten festgelegt werden. Die *neuen Komponenten* und die Integration in die beteiligten bestehenden Komponenten müssen konzipiert werden.

Für das Zusammenspiel der Rollen bzw. Komponenten müssen *Schnittstellen zur Wechselwirkung* (Steuerung) und zum *Informationsaustausch* erstellt werden. Dafür muss analysiert werden, welche

Informationen die Komponenten benötigen und es müssen *Datenmodelle* erstellt werden (Herkunft, Übertragung, Format bzw. Syntax und Semantik der Daten).

Es müssen (verteilte) *Algorithmen zur Pfadsuche* gefunden und analysiert werden, und nicht zuletzt müssen die Auswirkungen auf und *die Akzeptanz durch die Netzbetreiber* bei der Konzeption der Gateways berücksichtigt werden.

Das Gateway hat verschiedene Aufgaben, es muss Anfragen entgegennehmen und weiterleiten (Kommunikation mit anderen Gateways und Schnittstelle zu den Nutzern), es muss Pfade suchen, Ressourcen reservieren, Pfade schalten (d.h. auf Vermittlungsebene dafür sorgen, dass die Pakete einem Pfad folgen), es muss Verbindungszustände und Abrechnungsinformationen speichern und weitere Aufgaben erfüllen, die sich aus den beschriebenen Aufgaben ergeben. Dabei ist festzustellen, welche Aufgaben in jedem AS transparent behandelt, und welche Aufgaben inter-organisational gehandhabt werden können oder müssen. Aufgaben, die transparent behandelt werden können, sind beispielsweise die Reservierung von Ressourcen und die Umsetzung der Pfadschaltung. In diesen Fällen soll eine Schnittstelle definiert werden, da die Lösung von den unterschiedlichen Gegebenheiten in den autonomen Systemen abhängt (d.h. von den Mechanismen zur Reservierung von Ressourcen oder für die Pfadschaltung). Die Netzbetreiber können zu dieser Schnittstelle Backends für die eingesetzte Technik implementieren.

Eine zentrale Aufgabe, die Kommunikation zwischen den Gateways, kann offensichtlich nicht an die autonomen Systeme delegiert werden. Dafür muss in dieser Arbeit ein *Protokoll zur Pfadschaltungs-Kommunikation* mit und unter den Gateways entworfen werden, und in diesem Zusammenhang auch der *Austausch von QoS-Parametern* behandelt werden. Die QoS-Parameter müssen einerseits zwischen den autonomen Systemen ausgetauscht und verstanden werden, andererseits sind nicht überall die gleichen Parameter umsetzbar oder bekannt. Um dieses Dilemma zu lösen, sollen in dieser Arbeit einige *essentielle QoS-Parameter ausgesucht* werden, die jedes Gateway verstehen muss. Für deren Austausch müssen Syntax und Semantik festgelegt werden, der Gebrauch weiterer QoS-Parameter soll möglich sein. Damit die Netzbetreiber weitere beliebige QoS-Parameter verwenden können, müssen Möglichkeiten skizziert werden, diese zu definieren und deren Syntax und Semantik unter den autonomen Systemen bekannt zu machen. Die Schnittstelle zu den Verbindungsnutzern kann jeder Netzbetreiber eigenständig gestalten. In der Arbeit sollen Vorschläge zu verschiedenen Möglichkeiten gemacht werden.

Für die Pfadsuche sind *Routing-Informationen* und die *Ressourcenverfügbarkeit* notwendig, dafür sollen *Schnittstellen zu Routingprotokollen und Monitoring* skizziert werden.

Bei den Algorithmen zur Pfadsuche verhält es sich ähnlich wie bei den QoS-Parametern. Da die Pfadsuche auf Gateways verteilt stattfindet, ist es nicht möglich, einen einheitlichen Suchalgorithmus durch das Protokoll zu erzwingen.

Alle Lösungsdetails sollen auf ihre Auswirkung auf die Akzeptanz durch die Netzbetreiber hin untersucht werden. Das betrifft die Kommunikation zwischen den autonomen Systemen (Peering-Verträge, Abrechnung von Verbindungen, Austausch unterstützter QoS-Parameter, Bekanntmachung von Gateways), die technologischen Voraussetzungen (Notwendigkeit von Investitionen in neue Hardware, Umstellung auf andere Technologien), Risiko für bestehende Infrastruktur (Überlastung der Border-Router, neue Sicherheitslücken, schlechte Auslastung der Leitungen), finanzielle Anreize (Aussicht auf Einnahmen und Gewinne durch die Pfadschaltung) und die Skalierbarkeit (Testmöglichkeiten im kleinen Rahmen, problemloser Ausbau bei Erfolg).

1.3. Vorgehensweise

Aus der Aufgabenstellung ergeben sich bereits einige Anforderungen direkt. Um einen Bezug zum vorgesehenen Einsatz der Gateways herzustellen, werden in Kapitel 2 („Szenarien und Anforderungen“) einige Szenarien ausgewählt und die Anforderungen dieser Szenarien an die Pfadschaltung analysiert. Die Anforderungen aus den Szenarien und aus der Aufgabenstellung werden auf Relevanz und Kon-

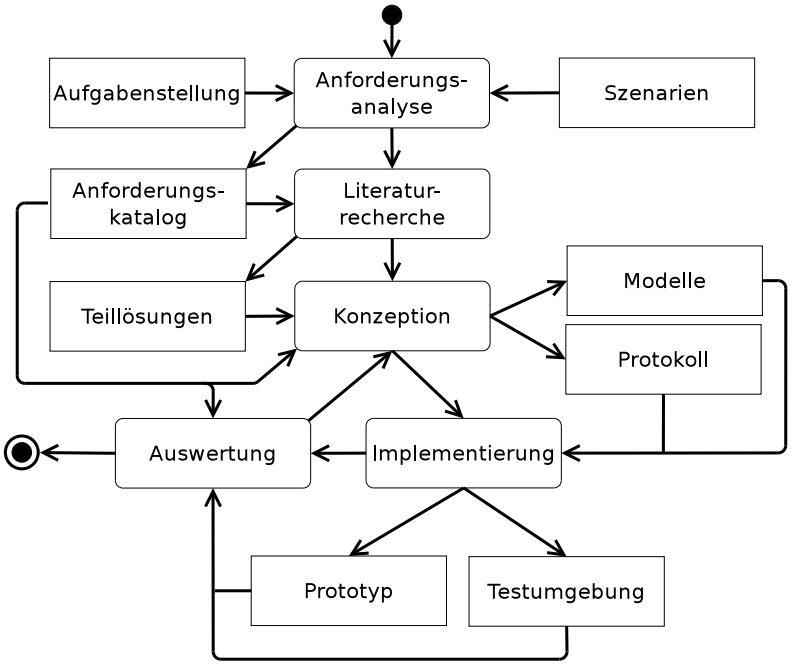


Abbildung 1.6.: Vorgehensweise

sistenz geprüft, nach Zugehörigkeit (Anforderungen an das AS, Anforderungen an das Protokoll etc.) sortiert und zu einem Anforderungskatalog zusammengefasst (Abschnitt 2.3).

Die Anforderungen werden mit dem Stand der Technik verglichen und existierende Teillösungen sowie Ideen für die Konzeption gesammelt. Im Kapitel 3 „Related Work“ werden verwandte Konzepte und Technologien auf ihre Verwendbarkeit für die Lösung untersucht und begründet, welche Teile davon ausgebaut oder neu konzipiert werden müssen, welche nicht geeignet oder benötigt sind, und wo noch keine Lösungen vorhanden sind.

Auf dem Hintergrund der Anforderungen und der bestehenden Konzepte wird die Konzeption der Gateways und der Pfadschaltung durchgeführt (Kapitel 4 „Konzeption des WAN-Gateway“) und ein Protokoll für die Pfadschaltung entworfen (Kapitel 5 „Das Path-Parameter-Control-Protocol“). Zum Testen und zur weiteren Ausarbeitung der Konzepte und des Protokolls wird eine Testumgebung mit virtuellen Maschinen erstellt und ein prototypisches Gateway implementiert (Kapitel 6 „Prototypische Implementierung“).

Abschließend wird die Konzeption und die prototypische Umsetzung ausgewertet und auf Probleme und Erweiterungsmöglichkeiten hingewiesen (Kapitel 7 „Fazit und Ausblick“).

1.4. Ergebnisse

Das Hauptergebnis dieser Arbeit ist ein Konzept für die Pfadschaltungs-Gateways und ihre Kommunikation über das *Path-Parameter-Control-Protocol (PPCP)*. Die Gateways können Netze mit nicht kompatiblen QoS-Technologien und QoS-Konzepten verknüpfen und ermöglichen die konsistente Verwendung dieser QoS-Technologien bei der Schaltung von Ende-zu-Ende-Pfaden. Die Pfadschaltung erfordert keine Eingriffe in das Routing-Modell des Internets und fügt sich ohne Erweiterungen in das existierende Routing ein.

Das Protokoll erlaubt die Formulierung eigener QoS-Parameter und die Definition von QoS-Parameter-Sets zur abstrakten Identifikation der QoS-Anforderungen spezifischer Verbindungstypen. Es spezifiziert aber auch eine Grundmenge an integrierten QoS-Parametern (Übertragungsrate, Latenz, Jitter

und Paketverlustrate) und berücksichtigt ihre Umsetzung durch prinzipiell unterschiedliche QoS-Verfahren (Priorisierung und Reservierung).

Bei der Verwendung von PPCP stehen Strategien zur Verfügung, die die Pfadsuche in einfachen Topologien durch (fast) gleichzeitige Pfadsuche und Pfadschaltung optimieren, für komplexe oder unbekannte Topologien ist aber auch eine ressourcenschonende Breitensuche möglich.

Das Protokoll wurde in einer Testumgebung aus virtuellen Maschinen erfolgreich getestet, die Pfadschaltung wurde auf den Gateways mit Linux und IP-Tables umgesetzt. Für die Umsetzung der QoS-Parameter wurden Schnittstellen zur Ressourcenverwaltung skizziert, die von der eingesetzten QoS-Technologie abstrahieren.

Für die Sicherheit und Abrechnung der Verbindungen werden Vorschläge gemacht und konzeptionell in PPCP integriert, um ihre Umsetzung vorzubereiten und zu vereinfachen.

2. Szenarien und Anforderungen

In diesem Kapitel wird eine Auswahl von Nutzungsszenarien vorgestellt, anhand derer die Facetten der dienstgütespezifischen Pfadschaltung erläutert werden. Aus diesen Facetten und aus der Analyse der Problemstellung sowie der Funktionsweise und Organisation des Internets wird ein Katalog mit Anforderungen an die Pfadschaltung und an die Gateways erstellt. Darauf aufbauend werden die Anforderungen, die die Pfadschaltung an die autonomen Systeme und die Netzbetreiber stellt, skizziert.

2.1. Szenarien

Die Auswahl der Nutzungsszenarien, die in den folgenden Abschnitten vorgestellt werden, ist einerseits durch deren Verbreitung, andererseits durch deren Relevanz für die Anforderungen motiviert. Trotz der teilweise weiten Verbreitung können die QoS-Probleme der Szenarien noch nicht als gelöst gelten.

2.1.1. Szenario „Video-on-Demand“

In diesem Szenario will ein Kunde einen Film eines Video-on-Demand-Anbieters über das Internet ansehen. Der Video-on-Demand-Anbieter will sicherstellen, dass der Film störungsfrei (ohne Ruckeln, Pausen oder Qualitätseinbußen) übertragen wird, und benötigt dafür eine Verbindung zum Kunden, die eine spezifische Übertragungsrate garantiert. Da das Abspielen des Films keine zeitkritische Interaktion mit dem Server erfordert (Pausieren und Fortsetzen des Films kann client-seitig gesteuert werden), ist die Latenz nicht so wichtig. Für die Implementierung eines Puffers spielt aber der QoS-Parameter „Jitter“ eine Rolle: Je weniger die Laufzeit der Video-Pakete variiert, umso kleiner kann der Puffer gewählt werden (wenn die Übertragungsrate konstant ist). Die Paketverlustrate muss berücksichtigt werden, wie hoch sie sein darf, hängt u. a. vom eingesetzten Video-Codec ab. Der Video-on-Demand-Anbieter will das Verbindungsmanagement für den Kunden transparent machen und den Verbindungsaufbau automatisieren. Dazu kann er ausnutzen, dass die Verbindungsdauer durch das Ansehen des Films festgelegt ist, und die Verbindung beim Anfordern des Films aufbauen und nach der Übertragung des Films wieder abbauen. Die Verbindungsauslösung findet hier also nicht durch Interaktion eines Menschen, sondern eines (Video-on-Demand-)Servers mit einem Gateway statt. Der Video-on-Demand-Server authentisiert sich bei einem Gateway und sendet einen Verbindungswunsch mit für dieses Video (abhängig von der Qualität) passenden QoS-Parametern und der Adresse des Endkunden (des Zuschauers) direkt an das Gateway. Nach erfolgtem Verbindungsaufbau sendet das Gateway die zugehörige Adresse zur unidirektionalen Nutzung an den Server, der die Daten ab sofort über diese Adresse überträgt. Der Anbieter möchte nicht, dass durch den Verbindungsaufbau eine Verzögerung verursacht wird, sondern gleichzeitig mit der Übertragung beginnen und die Verbindung auslösen. Sobald die Verbindung aufgebaut ist, soll nahtlos von normaler Übertragung auf Übertragung über die Verbindung umgestellt werden. Der Wechsel zur Übertragung über die Verbindung soll für Kunden und Server transparent sein (siehe Abbildung 2.1).

Die Zusammenfassung der Diskussion des Szenarios „Video-on-Demand“ ergibt folgende Facetten der Pfadschaltung:

- Berücksichtigung der QoS-Parameter „Übertragungsrate“, „Jitter“ und „Paketverlustrate“ (Paketverlust wegen Überlastung oder Verfälschung des Inhalts)
- Transparenz der Verbindung auf Client-Seite

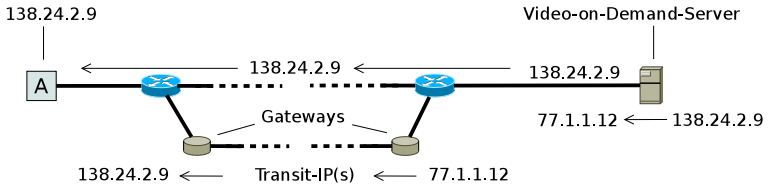


Abbildung 2.1.: Für den Endnutzer transparente Verbindung für Video-on-Demand

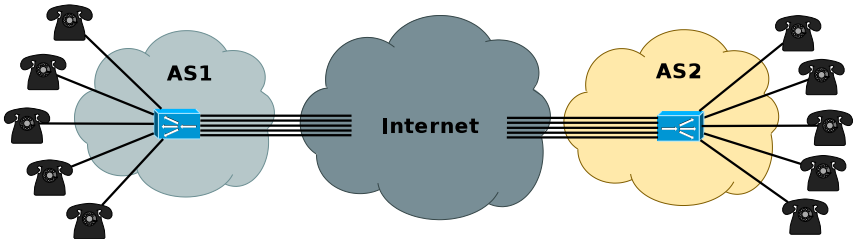


Abbildung 2.2.: Kooperation zwischen AS1 und AS2 für Ferngespräche über das Internet

- Transparenz der Verbindung auf Dienstseite
- Automatische Verbindungsauslösung (Authentisierung und Autorisierung)

2.1.2. Szenario „VoIP“

Im diesem Szenario wollen zwei Telefongesellschaften eine Kooperation für günstige Ferngespräche abschließen und die Daten für die Gespräche über das Internet übertragen (Abbildung 2.2). Für die Verfügbarkeit einer einzelnen Telefonverbindung kann nicht derselbe Aufwand betrieben werden, wie im Szenario „Videokonferenz“. Endverbraucher wollen günstige Telefontarife, für sie ist die gesonderte Behandlung einzelner Telefonverbindungen zu aufwändig. Auch ohne Garantien und Reservierung von Ressourcen in Routern und auf Schicht 2 kann QoS für viele Anwendungen zufriedenstellend funktionieren, das zeigt z. B. die Beliebtheit von Skype. Das lässt die Frage aufkommen, ob die Pfadschaltung für VoIP benötigt wird. Aus zwei Gründen ist sie trotzdem sinnvoll: Einerseits ist die Qualität bei VoIP noch verbesserungswürdig, besonders bei hoher Netzauslastung oder räumlicher Entfernung der Gesprächspartner, also in Fällen, die von der QoS-Pfadschaltung profitieren würden, und andererseits liegt die Kontrolle der Verbindungsqualität, wenn sie durch die Pfadschaltung erreicht wird, mehr beim Netzbetreiber und weniger bei Anwendungen, die selbstständig Pfade durch das Internet schalten (z. B. Peer-to-Peer-Anwendungen wie Skype), möglicherweise auf Kosten von Anwendungen, die der Netzbetreiber gerne priorisieren würde. In diesem Szenario geht es aber um die Sprachübertragung vieler Gespräche über einen Kommunikationskanal, nicht um einzelne VoIP-Verbindungen. VoIP mit eigener Verbindung für Endnutzer hat ähnliche, aber bescheidenere Anforderungen als das Szenario „Videokonferenz“ (Abschnitt 2.1.3). Die Gespräche könnten an den Enden des Kanals in ein Telefonnetz eingespeist werden oder an VoIP-Telefone bzw. Soft-Phones vermittelt werden.

Telefonverbindungen sollen störungsfrei und in guter Qualität möglich sein, allerdings sind die wenigsten Kunden bereit, für eine Verfügbarkeitsgarantie zu bezahlen. Kurze Störungen (wie beispielsweise im Mobilfunk) werden dagegen in Kauf genommen. Daher ist es in diesem Szenario sinnvoll, den Kommunikationskanal zu beobachten und bei einem Leistungseinbruch zu reagieren und einen weiteren Kanal zu schalten oder die Last zu verringern. Letzteres kann durch Unterbrechung der Gespräche, durch Ablehnung neuer Gespräche (Blockierung) oder durch transparente Umleitung von Gesprächen auf einen anderen Kanal geschehen. Diese Vorgehensweise kommt also ohne explizite Reservierung von Ressourcen aus und arbeitet stattdessen mit Priorisierung und Monitoring.

Den Paketen, die zu dem VoIP-Kommunikationskanal gehören, kann eine *höhere Priorität* zugeordnet werden, sodass sie die Router beim Scheduling bevorzugt behandeln oder andere Routen auswählen

(z. B. DiffServ, [NBBB 98]). Das verbessert Laufzeit und Paketverlustrate, Garantien können aber nicht gegeben werden (siehe auch Kapitel 3). Durch die Pfadschaltung kann einerseits die Prioritätsmarkierung der Pakete leichter bewerkstelligt werden (alle VoIP-Pakete haben eine vom Gateway vergebene Empfängeradresse und werden dadurch identifiziert), und andererseits kann dieser Markierung auch vertraut werden, sowohl im Hinblick auf die Semantik als auch auf die Berechtigung, da die Markierung im eigenen Netz vorgenommen wurde. An Trust-Boundaries (siehe Abschnitt 1.1) muss allerdings IP-Spoofing und unberechtigte Prioritätsmarkierung verhindert werden (siehe Abschnitt 2.2.4). Wegen der fehlenden Reservierung von Ressourcen in diesem Szenario müssen die Gateways bei starker Netzauslastung (nicht durch den VoIP-Provider, aber möglicherweise durch andere Pakete derselben Priorität) reagieren, um die Einhaltung der QoS-Parameter zu gewährleisten, z. B. durch Änderung des Pfades oder der Priorität. Dazu ist eine Schnittstelle zum Monitoring auf der Seite der Gateways nötig.

Ähnlich wie im Szenario „Video-on-Demand“ wird hier der Kommunikationskanal nicht von Endkunden aufgebaut, sondern durch den VoIP-Provider, und bleibt bei wechselnden Telefongesprächen bestehen (stehende Verbindung). Das Verbindungsmanagement sollte automatisch durch die VoIP-Gateways geschehen und auf eigene Monitoring-Informationen (nicht die der Gateways) reagieren, d. h. bei steigender Anzahl der Telefonverbindungen neue oder bessere Kanäle anfordern. Die Gateways müssen Unterstützung für das automatische Management bieten, sinnvoll wäre hier die Berücksichtigung von Maximalwerten für Ressourcen und Kosten.

Hier sind die Facetten des VoIP-Szenarios noch einmal zusammengefasst:

- Priorität statt Reservierung („Better Effort“ statt Garantie)
- QoS-Parameter „Latenz“, „Übertragungsrate“ und „Paketverlustrate“
- Monitoring durch Gateways (Überwachung der Einhaltung von QoS-Parametern bei fehlender Ressourcenreservierung)
- Multiplexing auf Seite des Verbindungsnutzers
- Monitoring durch Verbindungsnutzer (Überwachung der Verbindungsauslastung bzw. der Anzahl der laufenden Telefongespräche durch den VoIP-Provider)
- Kommunikation zwischen den Gateways zum Update der QoS-Parameter einer stehenden Verbindung
- Schnittstelle zwischen Verbindungsnutzer und Gateways für automatisches Verbindungsmanagement auf Grundlage der Auslastung (automatisches Auslösen einer Verbindung, Buchung von anderen QoS-Parametern oder Garantien bei laufender Verbindung)
- Einfluß der Preisgestaltung auf QoS („Billig-QoS“ ohne Reservierung von Ressourcen)

2.1.3. Szenario „Videokonferenz“

Eine Videokonferenz wird im geschäftlichen Umfeld immer öfter dazu eingesetzt, Reisekosten und Reisezeit zu sparen. Neben den für die Video- und Audioübertragung wichtigen QoS-Parametern „Übertragungsrate“ und „Latenz“ spielt bei der Konferenz auch eine garantierte *Verfügbarkeit* der Verbindung (inklusive ihrer Parameter) zu einer festgesetzten Zeit eine wichtige Rolle. Eine höhere Vermittlungspriorität der Verbindungspakete reicht dazu nicht aus (andere Pakete könnten die gleiche Priorität bekommen und Ressourcenengpässe verursachen), ein Blockieren wie bei Telefonanrufen kommt auch nicht in Frage, daher werden in diesem Szenario echte *Garantien* für die Verfügbarkeit der für die QoS-Parameter relevanten Ressourcen verlangt.

In Abbildung 2.3 ist eine Videokonferenz mit vier Standorten zu sehen. Zwei davon befinden sich in demselben autonomen System und sind über einen Router mit dem Internet verbunden. Zumindest bis zum ersten Router wäre *Multicast* für alle Standorte sinnvoll und würde die auf diesem Abschnitt benötigte Übertragungsrate dritteln. Für AS1 und AS2 wäre Multicast zu den zwei Standorten in AS3 ebenfalls sinnvoll, die Video- und Audiodaten aus AS1 und AS2 würden dadurch nicht doppelt

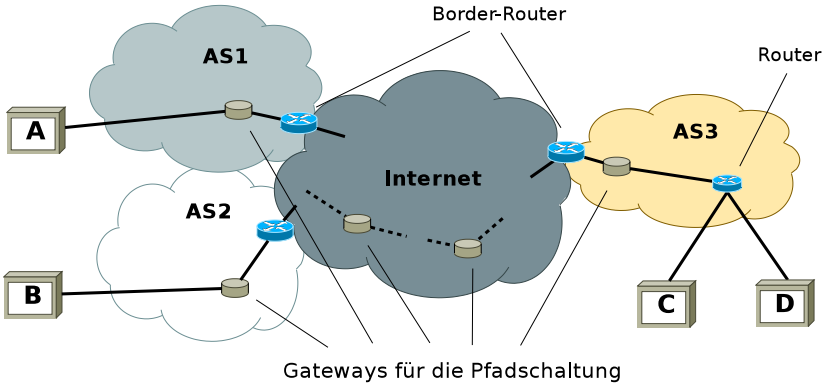


Abbildung 2.3.: Eine Videokonferenz mit vier Standorten, verteilt auf drei getrennte autonome Systeme

über das Internet übertragen. Ob Multicast auch in anderen Abschnitten der Verbindung sinnvoll ist, sieht man auf der Abbildung nicht, und auch in der Realität ist das meist nicht offensichtlich. Neben Multicast gibt es noch weitere Möglichkeiten für Synergien. Beispielsweise könnten sich die beiden Standorte in AS3 eine entsprechend dimensionierte Verbindung zu AS1 bzw. AS2 teilen (*Multiplexing*). Die Menge der zu übertragenden Daten würde zwar die gleiche bleiben, aber es müsste nur einmal ein Pfad gesucht werden, und es würden nur halb soviele Rewriting-Adressen benötigt. Eine geteilte Verbindung hat aber auch Nachteile. Es könnten beispielsweise zwei Verbindungen mit einfacher Übertragungsrate verfügbar sein, aber keine mit doppelter Übertragungsrate. Es könnte auch einer der beiden Standorte nicht fair spielen und versuchen, die geteilte Verbindung voll auszulasten.

Aus diesen Betrachtungen ergeben sich einige Konsequenzen für die Pfadsuche und -schaltung. Die Videokonferenz könnte einerseits über mehrere unidirektionale Verbindungen realisiert werden (jeder Standort fordert eine Verbindung zu den drei anderen Standorten an) und andererseits über eine einzige Verbindung (die dann nur von einem Standort angefordert wird und an allen Standorten verwendet wird, die anderen Standorte müssen möglicherweise über den erfolgreichen Verbindungsaufbau benachrichtigt werden). Hier stellt sich die Frage, wie die Identität einer Verbindung definiert wird, d. h. ob vier Bäume (d.h. vier Multicastadressen verknüpft mit je drei Pfaden) eine oder vier Verbindungen definieren. Die unidirektionalen Verbindungen bieten keine Möglichkeit für Synergien bei der Pfadsuche, da gar nicht klar ist, dass die Verbindungen zusammengehören, Multicast wäre aber möglich. Verwaltungstechnisch hat die Trennung der Verbindungen den Nachteil, dass jede Verbindung auch getrennt abgerechnet und einem Besitzer zugeordnet werden muss. Auch die Behandlung von Fehlern ist schwieriger: Fällt eine Verbindung aus oder kommt gar nicht erst zustande, so scheitert die Videokonferenz, und die drei anderen Verbindungen sind möglicherweise nicht mehr nötig und müssen einzeln abgebaut und, obwohl sie nicht genutzt werden konnten, bezahlt werden. Sowohl für die Abrechnung als auch für die Fehlerbehandlung ist es also vorteilhaft, die vier Multicastbäume als eine „*multidirektionale*“ Verbindung mit einem Auslöser bzw. Besitzer zu betrachten. Multidirektional heißt hier, dass die Absenderadressen „Teil der Verbindung“ sind, dass man sie also zur Kommunikation über die Verbindung verwenden kann. Die Pfadsuche wird durch die Zusammenfassung der vier Multicastbäume allerdings komplizierter. Will man die oben beschriebenen Synergien ausnutzen, so müssen die Möglichkeiten dafür bei der Pfadsuche erkannt werden. Dazu müssen Topologie und verfügbare Ressourcen analysiert werden (Abbildung 2.4: C und D teilen sich eine Verbindung zu B; für eine geteilte Verbindung zu A sind auf dem roten Abschnitt nicht genügend Ressourcen vorhanden, daher wird dieser Abschnitt nur von D genutzt, die Daten von A nehmen einen anderen Weg; auf dem letzten Abschnitt zu A werden die Pfade wieder zusammengeführt; letzteres ist automatisch der Fall, solange ein Ende nicht über mehrere IP-Adressen erreichbar ist; auf der gestrichelten Linie wäre Multiplexing möglich, allerdings ist für den Pfad zu A Demultiplexing, z. B. anhand der Absenderadresse, nötig; die Verfügbarkeit der technischen Möglichkeiten wie Demultiplexing muss bei der Pfadsuche genauso sichergestellt werden wie die Verfügbarkeit der Ressourcen zur Erfüllung der QoS-Parameter).

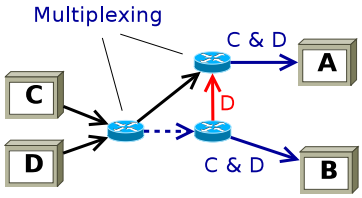


Abbildung 2.4.: Multiplexing bei *m:n*-Verbindungen

Synergien bei der Pfadsuche sind für die Netzbetreiber von Interesse, sie betreffen vor allem die Kommunikation der Gateways unterneinander. An Stellen, an denen Multiplexing in Frage kommt, muss sichergestellt werden, dass die Ressourcen korrekt (den QoS-Parametern entsprechend) aufgeteilt werden. Das kann einerseits dadurch geschehen, dass das Multiplexing von einem Scheduler ausgeführt wird, der die QoS-Parameter der einzelnen Verbindungen berücksichtigt, andererseits kann Multiplexing vom Verbindungsauslöser explizit erlaubt werden. In diesem Fall brauchen sich die Gateways nicht weiter um die Aufteilung der Ressourcen zu kümmern (z. B. weil der Auslöser garantieren kann, dass alle Enden die Nutzung der Ressourcen auf die Videokonferenzsoftware beschränken). Es sollten also zwei Arten des Multiplexing möglich sein, einmal die Variante, in der sich die Gateways nicht um die Aufteilung der Ressourcen kümmern müssen und die explizit vom Auslöser erlaubt werden muss, und einmal die Variante, die für die Endpunkte transparent ist und für die die Gateways sicherstellen müssen, dass Möglichkeiten zur korrekten Ressourcenaufteilung sowie zum Demultiplexing an allen relevanten Stellen vorhanden sind.

Für die multidirektionale „m zu n"-Verbindung muss bei der Pfadsuche definiert werden, wer Quelle und wer Senke ist und in welche Richtung welche QoS-Parameter gefordert sind. Soweit nicht, wie bei der Videokonferenz, überall die gleichen QoS-Parameter benötigt werden, müssen pro Verbindungszweig, also für je zwei Enden einmal pro Richtung, QoS-Parameter definiert werden.

Ebenso muss bei der Verbindungsauslösung definiert werden, ob es sich um Multicast- oder mehrere Unicastverbindungen handelt. In diesem Szenario sind alle Endpunkte sowohl Quelle als auch Senke, und jede Quelle sendet per Multicast an alle Senken (außer an sich selbst). Um die Komplexität im Rahmen des Notwendigen zu halten, werden Unicast- und Multicastverbindungen nicht gemischt, d. h. falls in diesem Szenario zwei Enden einen weiteren Kanal, z. B. für Dateitransfer während der Videokonferenz, benötigen, so muss dafür eine weitere Verbindung aufgebaut werden. Falls diese Verbindung kritisch für die Konferenzteilnehmer ist, trifft zwar auch hier zu, dass doppelt abgerechnet werden muss und dass die Konferenz nur dann zustande kommt, wenn beide Verbindungen aufgebaut werden, aber dieser Spezialfall rechtfertigt nicht eine signifikante Erhöhung der Komplexität. Der *m:n*-Verbindungsfall wird auf die Anforderungen dieses Szenarios beschränkt: Eine Teilmenge der Endpunkte sind Quellen und eine weitere (möglicherweise die gleiche) Teilmenge der Endpunkte sind Senken. Alle Quellen senden per Multicast an alle Senken. Eine Differenzierung der Senken in Abhängigkeit der Quelle muss demnach über mehrere Verbindungen geschehen.

Das Prinzip, nach dem Verbindungen bei [DKMY 11] und in dieser Arbeit realisiert werden, vereinfacht die Umsetzung von Multicast, da ohnehin ein Mechanismus zum Address-Rewriting benötigt wird. Dieser Mechanismus muss um eine Funktion zur Vervielfältigung von Paketen erweitert werden. Die Paketduplikate müssen dann an die verschiedenen Teilnehmer adressiert werden. Multicast wird also für normale Router (das sind nicht die, die für das Address-Rewriting zuständig sind) transparent und ohne Verwendung der Multicast-Adressbereiche umgesetzt. Wenn Multicast nicht zur Verfügung steht (weil kein Paketduplikator vorhanden ist), müssen mehrere Unicastverbindungen verwendet werden, was nicht nur zusätzliche Ressourcen, sondern auch zusätzliche Adressen verbraucht.

Die *m:n*-Verbindung besteht in diesem Szenario aus vier Multicastbäumen mit je einem Sender und drei Empfängern. Diese Multicastbäume könnten bi- bzw. multidirektional oder unidirektional sein. In diesem Zusammenhang bedeutet multidirektional, dass die Absenderadresse der Multicast-Pakete eine Verbindungsadresse (und nicht die tatsächliche IP-Adresse des sendenden Endpunktes) ist, sodass die Videokonferenz-Anwendung direkt auf die Pakete antworten kann; unidirektional bedeutet entsprechend, dass die Videokonferenz-Anwendung eine andere Adresse zum Senden per Multicast

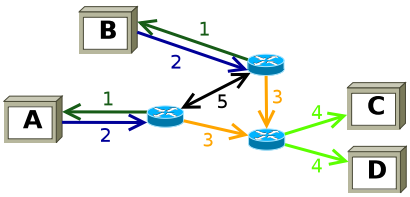


Abbildung 2.5.: Multidirektionalität bei *m:n*-Verbindungen, A und B senden

benötigt als die Absenderadresse der eintreffenden Pakete. Damit diese Absenderadresse zum Antworten verwendet werden kann, muss sie eine Multicastadresse sein, über die die drei anderen Endpunkte erreicht werden. Der Begriff „Absenderadresse“ bezieht sich hier also nur auf das Feld im IP-Header und nicht auf eine Adresse, über die ausschließlich der tatsächliche Absender erreicht wird. Hinter der Absenderadresse stehen also mehrere (hier drei) Absender bzw. Endpunkte. Anwendungen, die eine eindeutige Zuordnung der Absenderadresse zu einem Endpunkt benötigen (z. B. alle Anwendungen, die TCP verwenden, wobei bei Multicast TCP ohnehin nicht in Frage kommt), würden mit einer multidirektionalen Verbindung nicht mehr funktionieren. Abstrahiert man Multidirektionalität von diesem Szenario auf die oben beschriebenen *m:n*-Verbindungen, so gilt, dass jede Quelle einer *m:n*-Verbindung automatisch auch Senke ist (sonst ist „antworten“ nicht möglich), reine Senken als „Belauscher“ der Verbindung sind aber möglich, ohne die Bedeutung bzw. Verwendung der Begriffe zu beeinträchtigen. Eine allgemeine multidirektionale *m:n*-Verbindung enspricht also diesem Szenario mit zusätzlichen optionalen Lauschern. Die Videokonferenz-Anwendung in diesem Szenario könnte die Video- und Audiopakete der Absender nicht über die IP-Adresse, sondern anhand von UDP-Ports erkennen (nicht TCP-Ports, siehe oben), und über weitere Ports Steuerungskanäle schalten. Auf die Unidirektionalität bei Multicast kann, wegen möglicher Abhängigkeiten von eindeutigen IP-Absendern, nicht verzichtet werden. Andererseits bringen multidirektionale Verbindungen auch Vorteile: Die Anwendungen müssen nicht über Verbindungsadressen informiert werden, sie können einfach auf eintreffende Pakete antworten, und es muss nur eine Adresse (statt einer ausgehenden und drei eingehender Adressen) verwaltet werden. Für die Pfadsuche und das Address-Rewriting spielt die Multidirektionalität keine große Rolle, beim Address-Rewriting muss, mindestens auf dem letzten Gateway vor einem Endpunkt, zusätzlich zur Empfängeradresse auch die Absenderadresse ausgetauscht werden. Bei der Pfadsuche muss nur definiert werden, ob die Verbindung unidirektional ist oder nicht. Der Adress- und Ressourcenbedarf sowie die Anzahl der Stellen, an denen Address-Rewriting stattfindet, ist für beide Varianten gleich. Beim Einsatz von Multidirektionalität sind Vorsichtsmaßnahmen zu treffen, sonst kann es zu seltsamen Situationen kommen (Beispiel: Sendet A ein Ping an die Multicastadresse, antworten B, C und D jeweils der eigenen Multicastadresse, diese Antworten erhalten also auch Endpunkte, die kein Ping gesendet haben). Insbesondere sollte die Verbindung nicht versehentlich von Anwendungen genutzt werden, die für Punkt-zu-Punkt-Verbindungen gedacht sind (ping, traceroute, alle Anwendungen, die TCP verwenden etc.).

Abbildung 2.5 zeigt beispielhaft die multidirektionalen Multicastverbindungen aus Sicht von A und B (d.h. A und B senden je an die drei anderen Standorte). Auf den mit 1 beschrifteten Abschnitten ist die Absenderadresse die Multicastadresse (jeweils aus Sicht des Empfängers), auf den mit 2 beschrifteten Abschnitten hat diese Adresse die Rolle der Empfängeradresse („Antwort“ auf 1). Auf den mit 3 beschrifteten Abschnitten haben beide Verbindungen dieselbe Zieladresse, auf den mit 4 beschrifteten Abschnitten haben beide dieselbe Absenderadresse (die Multicastadresse je aus Sicht von C und D). Auf dem mit 5 beschrifteten Abschnitt müssen die Empfängeradressen für die Vermittlung zum nächsten Gateway geändert werden, die Absenderadressen können theoretisch an einem beliebigen Gateway geändert werden, solange sie nicht zur Vermittlung benötigt werden (z. B. bei Demultiplexing anhand von Absenderadressen).

Die Videokonferenz in diesem Szenario wird durch einen Administrator (den Verbindungsauslöser) in einem der Standorte koordiniert. Bevor die Übertragung beginnen kann, wird ein Verbindungswunsch an den ISP gesendet (beispielsweise per Mail). Der Verbindungswunsch definiert die IP-Adressen der 4 Standorte als Quelle und Senke der Verbindung und gibt die in alle Richtingen gleichen, zur störungsfreien Video- und Audioübertragung notwendigen QoS-Parameter (maximale Latenz, garantierte

minimale Übertragungsrate pro Standort und garantierte maximale Paketverlustrate) und weitere Verbindungsparameter (Multidirektionalität, Multicast, Verfügbarkeit und Anforderung garantierter Ressourcen) an. Nach erfolgter Pfadschaltung benachrichtigt der ISP den Admin und übermittelt vier Multicastadressen, eine für jeden Standort (gewöhnliche IP-Adressen, die hier als Multicastadressen dienen, keine Adressen aus dem von IP für Multicast vorgesehenen Adressbereich). Wäre die Verbindung nicht multidirektional, so müsste der Admin die anderen Standorte über die Adressen informieren, in diesem Fall genügt es aber, eine Anwendung zu konfigurieren, die anderen Standorte können auf die Initialisierung einfach antworten. Um die eintreffenden Pakete einem Video- bzw. Audiostream zuordnen zu können, kann jeder Standort einen anderen UDP-Source-Port verwenden, der bei der Initialisierung durch die Videokonferenz-Anwendung ausgehandelt werden könnte. Nach der Konferenz wird die Verbindung durch den Admin beendet, der ISP bestätigt den Verbindungsabbau und berechnet die Kosten.

Zusammengefasst ergeben sich aus der Diskussion des Szenarios „Videokonferenz“ die folgenden Facetten der Pfadschaltung:

- Berücksichtigung der QoS-Parameter „Übertragungsrate“, „Latenz“ und „Paketverlustrate“ (Paketverlust wegen Überlastung oder Verfälschung des Inhalts)
- Verfügbarkeit einer Verbindung
- Garantierte Ressourcen für eine Verbindung
- Optional Multicast
- Multiplexing von Zweigen einer Verbindung zwischen Gateways durch QoS-gesteuerte Aufteilung der Ressourcen
- Möglichkeit zum Austausch von QoS-Features
- Multiplexing durch Erlaubnis des Auslösers (ohne QoS-gesteuerte Ressourcenteilung durch die Gateways)
- Demultiplexing in Gateways
- Verbindung m:n, insbesondere n:n
- Option Unidirektionalität und Multidirektionalität (im Fall von m zu n: Simulation von Bidirektionalität durch Zusammenfassung aller Quellen – abzüglich einer – als Multicastsenke, die mit der übrigen Quelle bidirektional kommunizieren)
- Eindeutige Feststellbarkeit von Verbindungsidentität und Verbindungsbesitzer
- Schnittstelle zwischen Endkunden und ISP für Verbindungsauslösung und Beendigung durch Endkunden
- Schnittstelle zwischen ISP und Endkunden zur Benachrichtigung über erfolgten Verbindungsaufbau und Übermittlung der Verbindungsadressen zur Nutzung der Verbindung durch den Endkunden

2.2. Anforderungsanalyse

In diesem Abschnitt werden die Anforderungen an die Pfadschaltung von verschiedenen Seiten betrachtet und aus den Facetten der Szenarien Anforderungen gewonnen. Die analysierten Seiten der Pfadschaltung sind *Akzeptanz durch die Netzbetreiber* (Abschnitt 2.2.1), *Pfadschaltung* (Abschnitt 2.2.2), *QoS-Management* (Abschnitt 2.2.3) sowie *Sicherheit und Abrechnung* (Abschnitt 2.2.4).

2.2.1. Akzeptanz durch die Netzbetreiber

Die Anforderungen der Netzbetreiber sind denkbar einfach: Die Pfadschaltung soll bei wenig Risiko und Aufwand viel Nutzen und Gewinn bringen. Weiterhin ist es wichtig, den autonomen Systemen ihre Autonomie zu lassen; fremder Zugriff auf eigene Router und Ressourcen zur ferngesteuerten Pfadschaltung würde sicherlich auf breite Ablehnung stoßen. Die Risiken im Zusammenhang mit dienstgütespezifischer Pfadschaltung sind zahlreich. Die unkontrollierte oder ungewollte Reservierung von Ressourcen kann zu Engpässen und Ausfällen (DoS) führen, aber auch eine beabsichtigte Reservierung bzw. Nutzung von Ressourcen kann unvorhergesehene und ungewollte Folgen für den übrigen Datenverkehr haben. Bei der Pfadschaltung ist es unumgänglich, in das Routing einzugreifen, auch hier besteht das Risiko von ungewollten Änderungen (Subnetze werden ins Nichts oder über eine „teure“ Leitung geroutet). Das finanzielle Risiko will ebenfalls berücksichtigt sein, notwendige Investitionen sollten gering und skalierbar gehalten werden und die Nutzung vorhandener Infrastruktur möglich sein. Je nach Diversität ist damit aber auch ein erhöhter Aufwand für die Einbindung nötig, da für jede QoS-Technologie ein Backend nötig ist. Die Gateways sollten einfach und günstig zu testen sein, beispielsweise zuerst in Bereichen mit einfachen QoS-Backends und in kleinem Rahmen. Neben der neuen Technik müssen auch neue Geschäftsmodelle entwickelt werden, die in Abschnitt 2.1 vorgestellten Szenarien könnten sich auch dafür eignen. Sind Geschäftsmodelle gefunden und ist der Test im kleinen Rahmen erfolgreich verlaufen, so kommt es auf den einfachen Ausbau und die Skalierbarkeit an. Die Skalierbarkeit der Pfadsuche hängt von der Funktionsweise der Gateways ab und muss bei der Konzeption berücksichtigt werden. Der topologische Ausbau hängt aber auch wesentlich von der Verfügbarkeit von QoS-Backends ab, die vor ihrer Integration entwickelt und getestet werden müssen. Die QoS-Möglichkeiten, also die Verfügbarkeit spezifischer QoS-Parameter, hängen von den QoS-Backends ab. Um über unterschiedliche Backends hinweg einheitliche QoS-Parameter zu ermöglichen, ist eine geeignete Abstraktion der QoS-Parameter notwendig, die überall verfügbare und verbreitete Parameter definiert, aber die Verwendung der nicht überall verfügbaren, spezielleren Parameter nicht einschränkt.

Die Verbindungen über AS-Grenzen hinweg erfordern Abrechnungsmöglichkeiten, zum einen zwischen Netzbetreiber bzw. ISP und Kunde und zum anderen zwischen dem ISP und den Betreibern der Transitnetze. Dafür muss es beim Verbindungsaufbau möglich sein, die Verbindung einem Kunden und einem Netzbetreiber zuzuordnen. Der Kunde zahlt für die Verbindung bei seinem Provider, und der Provider vergütet den Verbindungsaufwand in den Transitnetzen mit einer Gegenleistung. Das kann durch erweiterte Peering-Agreements passieren, aber diese und die Policies, die das Inter-Domain-Routing regeln, sind heute schon sehr komplex – eine weitere Erhöhung der Komplexität zu vermeiden, wäre für die Akzeptanz einer Lösung sehr wichtig. Daher soll ein Abrechnungsmodell möglich sein, das auch unabhängig von Peering-Agreements bzw. den Verträgen zwischen den Netzbetreibern funktioniert, wie im folgenden Beispiel:

Ein Anbieter von QoS-Verbindungen bezahlt Ressourcen bzw. Zugriff auf dieselben in einem AS und bietet sie über ein dort betriebenes Gateway zur Pfadschaltung an, ohne dass der AS-Betreiber davon etwas davon weiß oder es in Peering-Verträgen berücksichtigt werden muss. Die Komplexität der Routing-Policies spiegelt sich in der Größe der Routingtabellen in den Border-Routern wider, entsprechend soll auch die Pfadschaltung selbst keine Zusatzbelastung für die Border-Router mitbringen, insbesondere darf nicht die Notwendigkeit entstehen, spezifische neue Features in die Border-Router zu integrieren (das schließt nicht aus, dass das möglich oder nützlich sein kann für die Pfadschaltung, es darf nur nicht notwendig sein). Hier sind die Anforderungen von und durch die Netzbetreiber zusammengefasst:

- Risikomanagement
- Erhalt der Selbstständigkeit der autonomen Systeme (keine fremden Eingriffe in Routing und Ressourcenverwaltung)
- Einfacher Test im (auch finanziell) kleinen Rahmen
- Einbindung vorhandener Infrastruktur und Technologie
- Skalierbarkeit

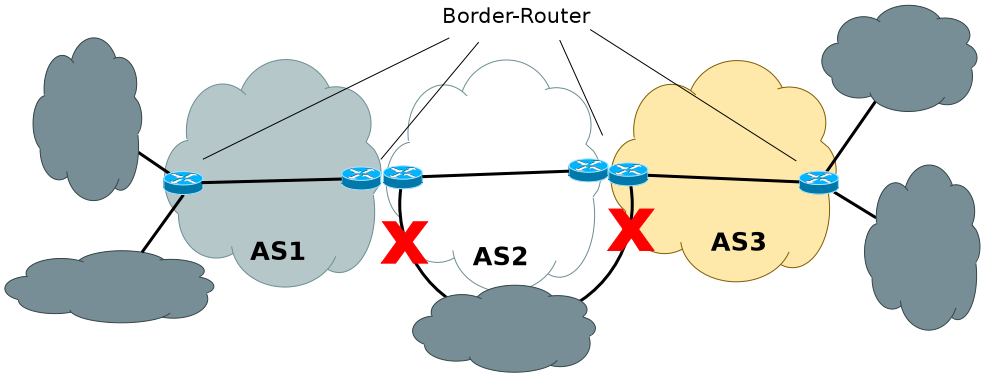


Abbildung 2.6.: Deterministisches Routing ohne Eingriffe in das Inter-Domain-Routing ist nicht nur zu benachbarten autonomen Systemen möglich

- Aussicht auf Einnahmen aus der Pfadschaltung, neue Geschäftsmodelle
- QoS-Backends und Transparenz der QoS-Technologie
- Abrechnungsmöglichkeiten zwischen Netzbetreibern und zwischen Netzbetreiber und Kunde
- Keine notwendige (Feature-)Änderung der Border-Router
- Keine notwendige Einbeziehung der Interdomain-Routing-Policies bzw. Peering-Agreements

2.2.2. Pfadschaltung

Um einen Pfad (siehe Abschnitt 1.1) im Internet zu schalten, der bestimmten QoS-Anforderungen genügt, muss das Routing soweit determinisiert werden, dass die Verbindung die für sie vorgesehenen Ressourcen auch nutzen kann. Ohne Eingriffe in das Routing ist es aktuell noch nicht einmal immer möglich, den Pfad festzustellen, den ein Paket im Augenblick nehmen würde (z. B. mit traceroute), geschweige denn, den Pfad zu bestimmen, den es in 5 Minuten nehmen wird. Für die dienstgütespezifische Pfadschaltung ist es unabdingbar, den Pfad zu kennen, den die Pakete einer Verbindung nehmen werden. Ohne Informationen über den Pfad können keine Ressourcen reserviert werden. Möglicherweise ist das auf dem Pfad, den die Pakete normalerweise nehmen könnten, auch gar nicht möglich, in diesem Fall müsste der Pfad geändert werden; auf jeden Fall muss er aber deterministisch sein.

Bei einem deterministischen Pfad erwartet jedes AS die Pakete einer Verbindung an bestimmten Border-Routern und leitet sie an ebensolche Border-Router des nächsten autonomen Systems weiter. Ohne Eingriffe in das Inter-Domain-Routing (und diese Eingriffe wurden oben ausgeschlossen) kann das nur durch Adressierung der Pakete in ein deterministisch erreichbares AS (d.h. ein AS, das nur über genau einen Pfad im Sinne von Abschnitt 1.1 erreichbar ist) geschehen. Das ist im Zweifelsfall ein benachbartes AS. Wenn das Gateway die Netztopologie aber genauer kennt, kann es auch entfernte autonome Systeme identifizieren, die nur über eine feststehende Kette von autonomen Systemen erreichbar sind. Das Routing wäre in diesem Fall deterministisch, wenn die Pakete an eine Adresse aus einem solchen entfernten AS adressiert werden (Abbildung 2.6: Pakete an von AS1 zu AS3 werden deterministisch geroutet, wenn AS1 sie über AS2 routet und von AS2 nach AS3 keine indirekte Route möglich ist).

Die Adressierung in das deterministisch erreichbare AS wäre durch „IP-over-IP“ möglich (die kompletten IP-Pakete werden in IP-Paketen transportiert, die an eine Adresse im nächsten deterministisch erreichbaren AS liegen, dort werden die Pakete aus- oder umgepackt analog zu z. B. IPSec im Tunneling mode, [KeSe 05]). Allerdings ist hier die individuelle Erfüllung von QoS-Parametern schwierig, da die Pakete aller Verbindungen dieselbe Zieladresse bekommen. Das Routing einer einzelnen Verbindung entlang bestimmter Pfade innerhalb des AS anhand der IP-Adresse ist damit nicht möglich. Um das zu ermöglichen, ist es notwendig, jeder Verbindung eigene IP-Adressen zuzuordnen, die für

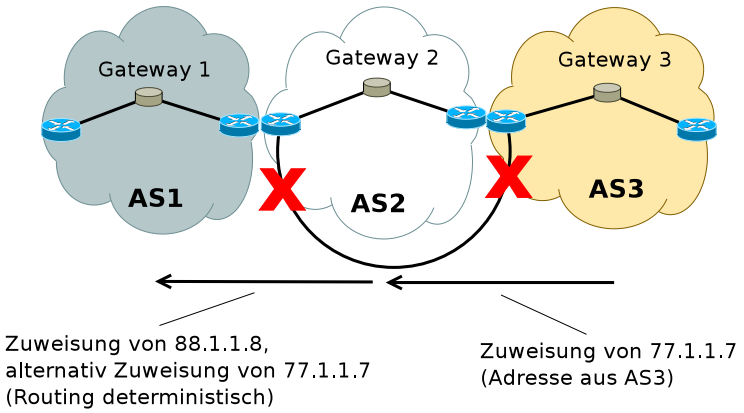


Abbildung 2.7.: Einer Verbindung wird von den Gateways eine Adresse aus einem deterministisch erreichbaren AS zugewiesen

das deterministische Routing sorgen. Eine eigene Adresse für jede Verbindung macht aber IP-over-IP überflüssig. Die eindeutige Zuordung macht es möglich, dass die zugewiesene Adresse direkt in den IP-Header eingetragen werden kann. Multiplexing (die Verwendung derselben IP-Adresse für das deterministische Routing mehrerer Verbindungen) ist dann immer noch möglich, solange die ursprüngliche Zieladresse wieder rekonstruiert werden kann (z. B. wenn die Absenderadresse feststeht und bekannt ist).

In Abbildung 2.7 weist Gateway 3 einer Verbindung die (zu AS3 gehörende) Adresse 77.1.1.7 zu und teilt diese Gateway 2 mit. Gateway 2 kann der Verbindung die (zu AS2 gehörende) Adresse 88.1.1.8 zuweisen und Gateway 1 mitteilen, welches dann alle Pakete der Verbindung an 88.1.1.8 adressiert. Wenn Gateway 2 sicher sein kann, dass die Adresse 77.1.1.7 von AS1 immer über AS2 nach AS3 vermittelt wird, kann es der Verbindung alternativ auch diese Adresse zuweisen und dadurch eine eigene Adresse einsparen. Innerhalb eines AS kann das Routing weiterhin statistisch passieren, solange das interne Routing für die benachbarten autonomen Systeme transparent ist und die für die Verbindung in diesem AS garantierten QoS-Parameter eingehalten werden.

Um Adressen für die Pfadschaltung verwenden zu können, müssen in den beteiligten autonomen Systemen *ungenutzte Adressen*, oder besser ein unbenutztes *Subnetz*, verfügbar sein. Das Routing dieser Adressen muss so angepasst werden, dass im eigenen AS QoS-Technologien zur Verfügung stehen und die an diese Adressen versendeten Pakete von außen an den richtigen Border-Routern eintreffen.

Für die Suche und Schaltung des Pfades muss ein neuer Dienst eingeführt werden, der diese Adressen verwaltet. Der Dienst soll auf einem beliebigen Knoten (dem Gateway, siehe Abschnitt 1.1) bereitgestellt werden. Es ist nicht ausgeschlossen, diesen Dienst auf Border-Routern laufen zu lassen, allerdings sind diese, wie oben angesprochen, stark ausgelastet und aufgrund ihrer Spezialisierung (spezielle Betriebssysteme IOS-XR/QNX, JUNOS) nur bedingt für die einfache Implementierung neuer Dienste geeignet. Ein Gateway muss in jedem autonomen System stehen, das an der Pfadschaltung beteiligt ist. Der Pfadschaltungsdienst muss zur Pfadsuche eine Zieladresse (den Endpunkt einer Verbindung) mit benachbarten autonomen Systemen verknüpfen, die Routing zu dieser Zieladresse anbieten. Aus diesen autonomen Systemen müssen diejenigen ausgesucht werden, die ebenfalls ein Pfadschaltungs-Gateway betreiben; diese Gateways können dann in die Pfadsuche einbezogen werden. Für diese Zuordnung von Zieladressen zu Nachbar-Gateways muss eine Informationsquelle bzw. eine Schnittstelle zu Inter-Domain-Routing-Informationen geschaffen werden. Für die QoS-Parameter einer Verbindung muss das Gateway Zugriff auf die Ressourcenverwaltung im eigenen AS haben, auch dafür ist eine Schnittstelle nötig. Wie im Abschnitt 2.2.4 beschrieben und begründet, ist ein Gateway nicht nur für die Pfadschaltung zuständig, sondern fungiert auch, stellvertretend für den Verbindungsnutzer, als Verbindungsbesitzer und -initiator. Für die Verbindungsauslösung werden Zugangsmechanismen zur Pfadschaltung bzw. zu den Gateways benötigt; einige sind in den Nutzungsszenarien in Abschnitt 2.1 vorgestellt. Dort, wo der Dienstnutzer direkt mit dem Gateway kommuniziert, ist die Beschränkung

auf einen einzigen Zugangspunkt sinnvoll. Sollte ein AS mehrere Gateways betreiben, muss die Auswahl des passenden Gateways automatisch bzw. durch den Netzbetreiber passieren. Dafür ist es nötig, einen Mechanismus zur automatischen Delegation von Verbindungsanfragen zu integrieren. Die Anforderungen aus diesem Abschnitt sind hier noch einmal zusammengefasst:

- Keine Eingriffe in das Inter-Domain-Routing
- Deterministischer Pfad durch Zuordnung von deterministisch erreichbaren Adressen
- QoS-Adresspool bzw. QoS-Subnetz in jedem an der Pfadschaltung beteiligten AS
- Anpassung des internen Routing an QoS-Parameter einer Verbindung
- Pfadschaltungsdienst auf den Gateways
- Ein einziger Zugangspunkt für Verbindungsnutzer
- Mechanismus zur automatischen Delegation von Verbindungsanfragen
- Schnittstelle vom Pfadschaltungsdienst zur QoS-Verwaltung
- Zugriff (lesend) vom Pfadschaltungsdienst auf Informationen zum Inter-Domain-Routing

2.2.3. QoS-Management

Aus den vorherigen Abschnitten ergeben sich einige Anforderungen an das QoS-Management. Die Festlegung auf eine bestimmte QoS-Technologie wurde bereits ausgeschlossen. In autonomen Systemen mit viel freien Kapazitäten ist es sinnvoll, andere (schwächere) Mechanismen zur Ressourcenverteilung zu wählen als in stark beanspruchten. Man könnte also noch weiter gehen und gar keine QoS-Technologie fordern – solange die zugesagten QoS-Parameter einer Verbindung auch eingehalten werden. Um letzteres zu erreichen, kann Monitoring eingesetzt und bei Überlastung eine Umleitung der Verbindung angefordert werden. Das ist überall dort sinnvoll, wo eine Verbindungsunterbrechung einer schlechten Verbindung vorzuziehen ist (z. B. im VoIP-Szenario, Abschnitt 2.1.2). Eine spezifische Verfügbarkeit kann damit natürlich nicht garantiert werden, dafür ist die Reservierung von Ressourcen und, je nach Art der Verfügbarkeit, Redundanz der Ressourcen nötig. Diese verschiedenen Arten der QoS-Sicherung und der mit ihnen verbundene Aufwand muss sich in der Formulierung der QoS-Parameter widerspiegeln können. Die Integration einer QoS-Beschreibungssprache, die beispielsweise eine differenzierte Formulierung der Verfügbarkeit einer Verbindung ermöglichen würde, hätte aber eine hohe Komplexität der Kommunikation zwischen den Gateways zur Folge. Die meisten Fälle würden von sehr speziellen QoS-Parametern und der damit einhergehenden Komplexität der Kommunikation gar nicht profitieren. Daher soll die in die Pfadschaltung integrierte QoS-Beschreibungssprache sich zum einen an weit verbreiteten QoS-Parametern orientieren und zum anderen an verbreiteten Technologien bzw. Methoden (siehe Monitoring) zur Umsetzung der QoS-Parameter, ohne den Einsatz komplexerer QoS-Beschreibungssprachen (und QoS-Technologien) zu verhindern. Es wäre auch möglich, gar keine QoS-Beschreibungssprache zu integrieren und es den Netzbetreibern zu überlassen, eine solche Sprache zu wählen bzw. zu entwickeln. Das würde aber einen einfachen Test und Einsatz verhindern und einen Vorteil der hier vorgestellten Pfadschaltung (Unabhängigkeit von lokalen Besonderheiten) relativieren. Die QoS-Parameter, die integriert werden sollen, wurden nach den Szenarien (Abschnitt 2.1) ausgewählt:

- minimale Übertragungsrate
- maximale Verzögerungszeit (Latenz)
- maximale Paketverlustrate
- maximale Verzögerungsvarianz (Jitter)

Zusätzlich ist noch eine Möglichkeit erforderlich, diese QoS-Parameter durch explizite Reservierung von Ressourcen zu sichern, damit die QoS-Parameter auch bei Überlastung garantiert werden können (wie im Szenario „Videokonferenz“, Abschnitt 2.1.3). Wird die explizite Reservierung von Ressourcen

nicht gefordert, so können die QoS-Parameter durch Monitoring bzw. Priorisierung (z. B. DiffServ) umgesetzt werden.

Da die Ressourcenverwaltung komplett in der Hand der Netzbetreiber liegt (und liegen soll), ist das Erzwingen einer Ressourcenreservierung bei den Nachbarn nicht möglich. Es muss darauf vertraut werden, dass die Nachbarn diese Ressourcen garantieren und die Verbindung nicht unterbrechen oder abbrechen, wenn die Reservierung von Ressourcen gefordert wurde. Die Abstraktion der QoS-Technologie bringt es mit sich, dass ein Nichteinhalten dieser Regeln mit anderen Mitteln verfolgt werden muss, z. B. auf finanziellem bzw. rechtlichem Weg oder durch Ausschluss der Gateways aus der Pfadsuche (Blacklisting). Durch Authentifizierung und Logging der Pfadschaltungsnachrichten kann die Pfadschaltung bzw. der Verbindungsaufbau rekonstruiert werden, die Einhaltung der Parameter einer laufenden Verbindung bei den Nachbarn ist hingegen schwieriger zu verfolgen. Dazu wäre es nötig, die Verbindung über mehrere autonome Systeme hinweg zu verfolgen, um das AS auszumachen, das die QoS-Anforderungen nicht erfüllt. Eine solche Überwachung einer beliebigen Verbindung würde die Grenzen der Pfadschaltung aber sprengen und sollte mit eigenen Werkzeugen gemacht werden. Was die Pfadschaltung ermöglicht, ist das „Testen“ eines einzelnen Verdächtigen. Dazu können zwei autonome Systeme, die keine gemeinsamen Grenzen miteinander, aber je eine gemeinsame Grenze mit dem Verdächtigen haben, einen Pfad durch das verdächtige AS schalten und durch Erzeugung von Datenverkehr die Einhaltung der QoS-Parameter testen. Liegt mehr als ein verdächtiges AS zwischen den Testern, oder wird der Datenverkehr nicht von den Testern (sondern z. B. von Kunden in einem anderen AS) erzeugt, ist es wesentlich schwieriger, die Einhaltung der QoS-Parameter zu überwachen, da keine Informationen von der Datenquelle und den autonomen Systemen dazwischen vorliegen. Testdaten über eine fremde Verbindung zu schicken ist auch keine Option (das würde der Verbindung Ressourcen wegnehmen und eine Filterung des Datenverkehrs erfordern, wenn die Testdaten nicht am Ende der Verbindung ankommen sollen), daher ist die Überwachung der Nachbarn keine Anforderung an die Pfadschaltung, die Überwachung der Verbindung innerhalb des eigenen Netzes aber schon.

QoS-Bedürfnisse von bestimmten Anwendungen können auch in Form von Erfahrungswerten vorliegen („diese Leitung ist bei exklusiver Nutzung für bis zu 100 simultane VoIP-Gespräche geeignet“). Werden regelmäßig Verbindungen benötigt, für die solche Erfahrungswerte (aber keine spezifischen und getesteten QoS-Parameter) vorliegen, so soll es möglich sein, unter den beteiligten autonomen Systemen diese QoS-Bedürfnisse zu kodieren, z. B. anhand einer QoS-ID (QoS-Set XY entspricht „Ressourcen für 100 simultane VoIP-Gespräche reservieren“). Damit Gateways Verbindungen für diese kodierten QoS-Parameter anbieten können, muss vorher getestet werden, welche Ressourcen dafür ausreichen und die ID des QoS-Parameter-Sets auf dem Gateway mit der Reservierung dieser Ressourcen verknüpft werden. Daraus ergibt sich die Anforderung, eine QoS-Set-ID in die Pfadschaltungskommunikation zu integrieren. Für autonome Systeme, die eine eigene QoS-Beschreibungssprache im Einsatz haben und QoS-Parameter in dieser Sprache direkt umsetzen können (z. B. durch Übergabe der QoS-Parameter an einen Ressourcenverwaltungsdienst), soll die Möglichkeit bestehen, Parameter in dieser QoS-Beschreibungssprache in die Pfadschaltungskommunikation einzubetten. Diese QoS-Parameter sollen von den Gateways direkt an das QoS-Backend weitergegeben werden und vor der Übertragung an das nächste Gateway „verpackt und gelabelt“ werden. Das nächste Gateway kann anhand des Labels entscheiden, ob es die Parameter versteht und gegebenenfalls die Parameter „auspacken“ und an das QoS-Backend weitergeben bzw. die Verbindung ablehnen. Das Gateway muss dabei nicht wissen, was die QoS-Parameter bedeuten und entsprechend auch nicht selbst feststellen, ob sie erfüllt werden können, diese Aufgabe wird an das QoS-Backend delegiert. Das QoS-Backend muss die QoS-Parameter bewerten und dem Gateway mitteilen, ob bzw. wie gut die QoS-Parameter erfüllt werden können. Das Gateway muss nur diese Information verarbeiten. Für die Art der Erfüllung von QoS-Parametern wäre eine Metrik wünschenswert, um den Gateways die Möglichkeit zu geben, differenzierte Entscheidungen zu treffen (z. B. QoS-Parameter können erfüllt werden, sind aber „teuer“).

Zusammenfassend ergeben sich die folgenden Anforderungen an das QoS-Management für die Pfadschaltung:

- Integration einer QoS-Beschreibungssprache für Standardfälle
- Berücksichtigung von Übertragungsrate, Latenz, Paketverlustrate und Jitter

- Möglichkeit zur Integration eigener QoS-Beschreibungssprachen bzw. zur Formulierung eigener QoS-Parameter
- Definition von QoS-Sets, die über eine ID angefordert werden
- Keine Festlegung bezüglich des Einsatzes von QoS-Technologie, aber
- Berücksichtigung grundsätzlicher Unterschiede der QoS-Sicherung (Reservierung/Verfügbarkeit vs Priorisierung/Monitoring)
- Flag, um Verfügbarkeit (Reservierung) von Ressourcen zu fordern
- Transparenz der QoS-Umsetzung aus Sicht der Pfadschaltung (höchstens die Gateways wissen, wie die QoS-Parameter umgesetzt werden, und das auch nur für das eigene AS)
- Metrik zur Bewertung der Erfüllbarkeit von QoS-Parametern (Erfüllung „günstig“ oder „teuer“)
- Monitoring und Ressourcenverwaltung durch die Netzbetreiber (durch eigenes Gateway, nicht durch fremde Gateways)
- Kein implizites Monitoring der Nachbarn (widerspricht der Selbstständigkeit der Netzbetreiber und sprengt den Rahmen der Pfadschaltung)

2.2.4. Abrechnung und Sicherheit

Eine Verbindung bindet Ressourcen und verursacht Kosten. Das Binden von Ressourcen impliziert die Gefahr von DoS-Angriffen, und die Verteilung der Kosten muss nachvollzogen werden können, daher ist es notwendig, die Auslösung und die Verwaltung einer Verbindung abzusichern. Dafür ist ein Mechanismus nötig, der den Absender der Pfadschaltungsnachrichten (der Nachrichten, die den Verbindungsauf- und -abbau und Verbindungsparameter steuern) authentifiziert und die Pfadschaltung autorisiert. Die Verwaltung einer kompletten Liste der Zugriffsberechtigten und der Art der Berechtigung, d.h. aller möglichen zur Verbindungsauslösung autorisierten Endnutzer in allen autonomen Systemen inklusive der jeweils zur Verfügung stehenden Parameter, ist enorm aufwändig und entsprechend wenig sinnvoll, daher muss die Authentifizierung der Pfadschaltungsnachrichten auf die Gateways beschränkt werden, d. h. Verbindungsauf- und -abbau sowie eine Verbindungsänderung kann nur durch ein Gateway erfolgen, das auch als Verbindungsbesitzer fungiert. Damit reduziert sich die Liste der Zugriffsberechtigten auf die Gateways oder sogar, abhängig von der Implementierung, auf benachbarte Gateways (d.h. die direkten Kommunikationspartner eines Gateways). Die Verwaltung der Verbindungsauslöser und Nutzer wird damit an die zuständigen Gateways bzw. die Gateway-Betreiber delegiert und kann entsprechend frei gestaltet werden. Damit ist die Autorisierung und Authentifizierung aufgeteilt in einen lokalen Teil und einen globalen Teil. Den lokalen Teil kann jeder AS-Betreiber nach Belieben gestalten; Beispiele, die in der Konzeption berücksichtigt werden sollten, sind in den Nutzungsszenarien beschrieben. Der globale Teil muss aber konsistent sein, da jedes Gateway die Möglichkeit haben muss, den Verbindungsbesitzer zu authentifizieren und ihm die Verbindungskosten in Rechnung zu stellen.

Zur Authentifizierung muss die Möglichkeit vorgesehen werden, die Pfadschaltungsnachrichten bzw. Teile davon zu signieren, außerdem wird ein Schutz gegen Replay-Angriffe benötigt (z. B. signierter Zeitstempel, Bestätigungs- oder stichprobenartige Challenge-Response-Mechanismen: Wenn diese Verbindung jetzt aufgebaut werden soll, authentifiziere dich durch Signierung dieses Token). Zur Identifikation einer Verbindung ist eine eigene und (zumindest lokal) eindeutige ID notwendig, da keine eindeutigen Identifikationsmerkmale unter den Verbindungseigenschaften zu finden sind (wie z. B. die zwei IP-Adressen und zwei Ports einer TCP/IP-Verbindung; Adressen, Besitzer, QoS-Parameter etc. können bei der Pfadschaltung auf mehrere Verbindungen passen). Zumindest diese ID und der Verbindungsbesitzer sollen global konstant und eindeutig (wenn der Verbindungsbesitzer eine ID nicht zweimal vergibt) sein, damit eine Verbindung darüber eindeutig identifizierbar ist. Da die anderen Teile der Nachrichten, insbesondere die QoS-Parameter, verfälscht werden können, muss sich auch der Nachrichtensender authentifizieren können.

Ein Zeitstempel ist nicht nur für die Gültigkeit einer Signatur (Replay-Schutz) sinnvoll, sondern auch für die zeitliche Beschränkung der Pfadsuche, daher wird die Möglichkeit eines Zeitstempels zumindest für einen Teil der Pfadschaltungsnachrichten gefordert. Ein Bestätigungsmechanismus, der auch vor Replay-Angriffen schützen könnte (Verbindungsaufbau zusätzlich vom Besitzer direkt und nicht nur vom Nachbar-Gateway bestätigen lassen), ist auch dazu geeignet, andere Betrugsversuche aufzudecken (z. B. könnte ein Netzbetreiber die Übertragungsrate vergrößern und die Verbindung für eigene Zwecke missbrauchen).

Ein wichtiger Punkt ist auch das IP-Spoofing. Einerseits kann durch Spoofing eine Verbindung „gekapert“ werden (wenn die Verbindungsadresse von Malice gesnifft und verwendet wird), andererseits ist die normale bzw. bisherige Anwendung von Anti-Spoofing-Maßnahmen problematisch: Wird bei einer bidirektionalen Verbindung eine Verbindungsadresse als Senderadresse verwendet, werden per default möglicherweise alle Pakete verworfen. Hier muss entweder eine Anpassung der Anti-Spoofing-Maßnahmen vorgenommen werden oder die Senderadresse erst nach der Anwendung der Anti-Spoofing-Maßnahmen geändert werden. Dieses Problem betrifft vor allem den Pfadabschnitt bzw. die Trust-Boundary zwischen Netzbetreiber und Kunden, und daher besonders die Anforderungen an die Kommunikation auf diesem Abschnitt. Hier sind die Punkte aus diesem Abschnitt, die bei der Konzeption berücksichtigt werden müssen, noch einmal zusammengefasst:

- Aufteilung von Authentifizierung und Autorisierung zur Pfadschaltung in lokalen und globalen Teil
- Authentifizierung und Autorisierung des Verbindungsauslösers nach den Szenarien in Abschnitt 2.1
- Authentifizierung der Pfadschaltungsnachrichten zwischen den Gateways
- Autorisierung der Pfadschaltung in Gateways
- Identifikation einer Verbindung durch eine Verbindungs-ID in Pfadschaltungsnachrichten
- Chain-of-Trust für Pfadschaltungsnachrichten (jeder muss zumindest seinen direkten Kommunikationspartnern vertrauen können)
- Signieren von globalen Teilen der Pfadschaltungsnachrichten durch den Verbindungsbesitzer
- Falls eine Signatur durch den Verbindungsbesitzer fehlt, Vertrauensdelegation an den Nachbarn und (implizite) Übernahme der Verantwortung für die Verbindung
- Signieren von temporären (lokalen) Teilen der Pfadschaltungsnachrichten (oder der kompletten Pfadschaltungsnachrichten) durch den Sender
- Zeitstempel für (manche) Pfadschaltungsnachrichten für die Gültigkeitsdauer (von Signaturen, Suchanfragen etc.)
- Optional: Schnittstelle zur Nutzung CA-signierter Webserver-Schlüssel (https)
- Optional: Stichproben der Pfadschaltungsnachrichten speichern oder vom Verbindungsbesitzer automatisch prüfen lassen
- Abrechnung anhand der Signaturen bzw. der Authentifizierung des Nachbarn oder Verbindungsbesitzers
- Missbrauch durch IP-Spoofing
- Paketverlust durch Anti-IP-Spoofing

2.3. Anforderungskatalog

Die Anforderungen und Facetten, die sich aus den vorherigen Abschnitten ergeben haben, werden hier zusammengefasst und kategorisiert. Zuerst werden die Anforderungen, die zur inter-organisationalen

Verknüpfung bereits vorhandener QoS-Möglichkeiten zu einem Pfad beitragen, anschließend die erweiterten Anforderungen, die szenariobedingt wünschenswerte zukünftige Möglichkeiten wie Multicast und „Multidirektionalität“ im Fokus haben, aufgelistet.

Die erweiterten Anforderungen werden in dieser Arbeit zwar in die Konzeption mit einbezogen, können aber, da die technischen Möglichkeiten dazu noch nicht ausreichend entwickelt sind oder weil sie außerhalb des Umfangs dieser Arbeit liegen, nicht oder nur unvollständig umgesetzt werden.

Anforderungen an die dienstgütespezifische Pfadschaltung und die WAN-Gateways

1. Schaltung eines (deterministischen) Ende-zu-Ende-Pfades durch das Internet auf IP-Ebene
2. Berücksichtigung der QoS-Parameter „Übertragungsrate“, „Latenz“, „Jitter“ und „Paketverlustrate“ (Paketverlust wegen Überlastung oder Verfälschung des Inhalts)
3. Austausch der QoS-Parameter (Festlegung von Syntax und Semantik)
4. Möglichkeit zur Mengenbildung von verbindungsspezifischen QoS-Parametern
5. Festlegung der Verfügbarkeit einer Verbindung durch Reservierung von Ressourcen
6. Option für alternatives QoS-Provisioning über Prioritäten (statt Reservierung)
7. Monitoring durch Gateways (Überwachung der Einhaltung von QoS-Parametern im eigenen AS bei Prioritäts-QoS)
8. Option für Unidirektionalität oder Bidirektionalität
9. Eindeutige Feststellbarkeit der Verbindungsidentität
10. Eindeutige Zuordnung einer Verbindung zu einem „Besitzer“
11. Update der QoS-Parameter einer bestehenden Verbindung
12. Transparenz der Verbindung an den Endpunkten
13. Konzeptionelle Aufteilung der Pfadschaltung in lokalen (Verbindungsauslöser zu Gateway) und globalen (Gateway zu Gateway) Teil
14. Möglichkeit zur Beschränkung der Gültigkeitsdauer von Pfadschaltungsnachrichten
15. Transparenz der QoS-Umsetzung aus Sicht der Pfadschaltung (Gateways wissen, wenn überhaupt, nur von der lokalen Umsetzung der QoS-Parameter)
16. Metrik zur Bewertung der Erfüllbarkeit von QoS-Parametern (Erfüllung „günstig“ oder „teuer“)
17. Ein einziger Zugangspunkt für Verbindungsnutzer (impliziert einen für den Verbindungsnutzer transparenten Delegationsmechanismus für Verbindungsanfragen)
18. Definition einer Schnittstelle vom Pfadschaltungsdienst zur QoS-Verwaltung
19. Konzeption einer Informationsquelle für die Zuordnung von Adressen (Netzen) zu benachbarten Gateways (Schnittstelle zum Inter-Domain-Routing)
20. Transparentes Multiplexing von Verbindungen zwischen Gateways durch QoS-gesteuerte Aufteilung der Ressourcen („gerechtes Multiplexing“)
21. Demultiplexing in Gateways
22. Schnittstelle zwischen Endkunden und ISP für Verbindungsauslösung und Beendigung durch Endkunden
23. Schnittstelle zwischen ISP und Endkunden zur Benachrichtigung über erfolgten Verbindungsaufbau und Übermittlung der Verbindungsadressen zur Nutzung der Verbindung durch den Endkunden

24. Schnittstelle zwischen Diensten (Video-on-Demand, VoIP) und Gateways für nutzungs- und lastgesteuertes Verbindungsmanagement
25. Möglichkeit zum Austausch spezifischer QoS-Features zwischen den Gateways
26. Möglichkeit zur Integration neuer QoS-Beschreibungssprachen bzw. neuer QoS-Parameter

Erweiterte Anforderungen unter Einbeziehung von Sicherheit, Abrechnung und Multicast

1. Konzeptionelle Aufteilung von Authentifizierung und Autorisierung zur Pfadschaltung in lokalen (Verbindungsauslöser zu Gateway) und globalen (Gateway zu Gateway) Teil
2. Möglichkeit zur automatischen Authentisierung des Verbindungsauslösers und Besitzers
3. Multicastverbindungen
4. Verbindung m zu n, insbesondere n zu n (Videokonferenz)
5. Option für Unidirektionalität und Multidirektionalität (siehe Szenario „Videokonferenz“, Abschnitt 2.1.3)
6. Multiplexing bei *m:n*-Verbindungen durch Erlaubnis des Auslösers (ohne QoS-gesteuerte Ressourcenteilung beim Multiplexing durch die Gateways, Ressourcenteilung durch Endpunkt)
7. Monitoring durch bzw. Feedback an den Verbindungsnutzer (Überwachung der Verbindungsauslastung bzw. der Anzahl der laufenden Telefongespräche durch den VoIP-Provider)
8. Authentifizierung des Absenders der Pfadschaltungsnachrichten
9. Autorisierung der Pfadschaltung in Gateways
10. Mechanismen gegen IP-Spoofing integrieren (Benachrichtigung über neue, „erlaubte“ Absenderadressen)
11. Stichproben der Pfadschaltungsnachrichten speichern oder (verpackt) an den Verbindungsbesitzer senden

Anforderungen an und durch autonome Systeme

Die Integration der Gateways in die autonomen Systeme ergibt, neben spezifischen, auch einige allgemeine Anforderungen, die bei der Konzeption der Gateways berücksichtigt wurden und sich teilweise in den oben genannten spezifischen Anforderungen widerspiegeln:

1. Risikomanagement
2. Erhalt der Selbstständigkeit der autonomen Systeme (keine fremden Eingriffe in Routing und Ressourcenverwaltung)
3. Einbindung vorhandener Infrastruktur und Technologie
4. Schnelle und einfache Umsetzung im (auch finanziell) kleinen Rahmen (Testen der Gateways und der Pfadschaltung)
5. Skalierbarkeit (Ausbaumöglichkeiten nach erfolgreichen Tests)
6. Etablierung neuer Geschäftsmodelle (QoS-Anbieter)
7. Implementierung von QoS-Backends zur Vermittlung zwischen Gateways und spezifischen QoS-Technologien
8. Abrechnungsmöglichkeiten zwischen Netzbetreibern und zwischen Netzbetreiber und Kunde
9. Keine notwendige (Feature-)Änderung der Border-Router

10. Keine notwendige Einbeziehung der Interdomain-Routing-Policies bzw. Peering-Agreements
11. Keine Festlegung bezüglich des Einsatzes von QoS-Technologie
12. Kein Monitoring bei den Nachbarn (widerspricht der Selbstständigkeit der Netzbetreiber)
13. Keine notwendigen Eingriffe in das Inter-Domain-Routing
14. QoS-Adresspool bzw. QoS-Subnetz(e) in jedem an der Pfadschaltung beteiligten AS
15. Anpassung des internen Routing an QoS-Parameter einer Verbindung

In diesem Kapitel wurden einige Szenarien vorgestellt, die bereits weit verbreitet sind. Ist diese Arbeit noch nötig, obwohl IP-Telefonie, Online-Games und viele andere Dienste, die QoS benötigen, bereits massenhaft im Einsatz sind? Im Einsatz zeigt sich, dass als „QoS-Ersatz“ spezielle Maßnahmen ergriffen werden müssen, die in der Regel nur begrenzt funktionieren. Eine dieser Maßnahmen, die beispielsweise bei Online-Games zum Einsatz kommt, ist die (räumliche) Verkürzung des Abstandes der Verbindungsendpunkte. Beliebte Online-Games stellen lokale Server zumindest auf jedem Kontinent zur Verfügung, aber auch da kommt es regelmäßig zu Engpässen. Sowohl die Entfernung zum Server als auch die Netzauslastung ist oft deutlich spürbar („Lag“). Bei Internettelefonie sinkt die Sprachqualität durch gleichzeitige intensive Internetnutzung spürbar und Gesprächsabbrüche sind trotz *Overprovisioning* keine Seltenheit, auch bei räumlicher Nähe der Verbindungsendpunkte (Telefone). Diese auf Best-Effort basierenden Ansätze sichern einer IP-Verbindung weder eine spezifische Qualität, noch lösen sie das Pfadschaltungsproblem. Skype ist ein gutes Beispiel für Best-Effort und dessen Probleme. Einerseits kann Skype Ressourcen „finden“ und nutzen, oft ohne Einwilligung bzw. sogar gegen den Willen der Provider, andererseits können diese den Skype-Verkehr gezielt sabotieren (üblich insbesondere in Handynetzen), da er in Konkurrenz zu eigenen Angeboten (den Mobiltelefon-Tarifen) steht. Die Schaltung einer Verbindung mit geeigneten QoS-Parametern und einem authentifizierten Besitzer kann, unter Einbeziehung der Endnutzer und der ISPs, beide Probleme lösen. Das Verhältnis der hier vorgestellte Lösung zu existierender Technik (unter anderem auch Skype) wird im folgenden Kapitel ausgeführt.

3. Related Work

Seit der Standardisierung des Internet-Protokolls in [Post 81] wurden viele Ansätze und Lösungen für QoS im Internet vorgelegt. Der IP-Standard definiert die Interpretation des Type-of-Service-Byte (ToS-Byte) im IP-Header zur Angabe der gewünschten Servicequalität (QoS), die Definition wurde durch Differentiated-Services (DiffServ: [NBBB 98], [BBC$^+$ 98] und [RFB 01]) aktualisiert.

DiffServ teilt IP-Pakete anhand des ToS-Byte in Klassen ein und erlaubt eine differenzierte Behandlung der IP-Pakete nach Klassenzugehörigkeit. DiffServ erlaubt keine Zuordnung von Ressourcen zu einer Ende-zu-Ende-Verbindung und garantiert keine spezifischen QoS-Parameter. QoS-Verfahren, die garantierte QoS-Parameter einer Ende-zu-Ende-Verbindung auf IP-Ebene ermöglichen, sind die Integrated-Services (IntServ: [Wroc 97b], [Wroc 97a] und [SPG 97]) in Verbindung mit dem Resource-Reservation-Protocol (RSVP und die Erweiterung RSVP-TE: [BZB$^+$ 97] und [ABG$^+$ 01]).

IntServ und RSVP setzen voraus, dass alle an einer Ende-zu-Ende-Verbindung beteiligten Knoten IntServ und RSVP unterstützen. Außerdem verursacht jede mit IntServ/RSVP geschaltete Verbindung erheblichen Verwaltungs-Overhead: Die Router müssen den Zustand jeder Verbindung speichern und regelmäßig entlang der Verbindung aktualisieren, wodurch der Ressourcenverbrauch auf den Routern und das Datenverkehrsaufkommen erhöht wird. Die Voraussetzungen für den Einsatz von IntServ und RSVP sind in der Realität oft nicht erfüllt.

Auf Schicht 2 des OSI-Modells ist mit ATM[1] eine QoS-fähige Technik verfügbar, für die Anwendung bei einer Ende-zu-Ende-Verbindung auf IP-Ebene müssen die ATM-Links aber untereinander zu einem Pfad verknüpft und das Routing auf diese Links festgelegt werden. Auch ist ATM nicht durchgehend verfügbar, die QoS-Fähigkeiten von ATM müssen mit anderen Techniken wie MPLS [RVC 01] [RTF$^+$ 01] (erweitert um QoS-Fähigkeiten durch CR-LDP [JAC$^+$ 02]) verknüpft oder ergänzt und um eine Ende-zu-Ende-Pfadschaltung erweitert werden.

Ebenfalls auf MPLS baut das in [AAC$^+$ 03] vorgestellte QoS-fähige Traffic-Engineering anhand von DiffServ auf, allerdings sind, bedingt durch DiffServ, keine QoS-Garantien für Ende-zu-Ende-Verbindungen möglich, und eine Beschränkung der Pfadschaltung auf MPLS/DiffServ widerspricht den Anforderungen an die Unabhängigkeit der Netzbetreiber aus Kapitel 2.

In [Roel 05] wird eine Methodik zur Koppelung von Netz-QoS-Architekturen beschrieben, Adressierung und Routing werden aber explizit ausgeklammert.

In [DKMY 11] liegt der Fokus dagegen gerade auf Adressierung und Routing. Dort wird vorgeschlagen, einen Pfad über Technologie- und AS-Grenzen hinweg auf IP-Ebene zu schalten, indem die IP-Pakete immer an ein Gateway im jeweils nächsten autonomen System adressiert werden. Die eingesetzten QoS-Verfahren sind den Netzbetreibern überlassen, solange die ausgehandelte Dienstgüte eingehalten wird. Viele der Ideen aus [DKMY 11] sind in diese Arbeit eingeflossen.

In [CNRS 98] werden unter anderem die Probleme von QoS-fähigem Routing über AS-Grenzen hinweg und die Beschränkung von RSVP beschrieben. Der dort gemachte Lösungsansatz zielt aber, anders als [DKMY 11], auf eine Erweiterung des Internet-Routing-Modells ab.

Protokolle für die Sitzungsverwaltung, wie das Session-Initiation-Protocol (SIP: [RSC$^+$ 02]), könnten von QoS-Fähigkeiten profitieren, indem sie QoS-Parameter passend zu einem Sitzungstyp anfordern (z. B. eine VoIP-Sitzung). Die Datenübertragung in SIP-Sitzungen geschieht über normales Routing, die feste Zuordnung von Ressourcen zu einer Sitzung und die garantierte Erfüllung von

[1] Eine Liste aller durch das ATM-Forum anerkannter ATM-Standards findet sich unter:
http://web.archive.org/web/20050715034550/http://www.atmforum.com/standards/approved.html

QoS-Anforderungen sind daher nicht möglich bzw. nur dann möglich, wenn sich alle Enden in einer Routing-Domain befinden (siehe z. B. [CNRS 98]).

SOCKS ([LGL+ 96]) bietet einen Proxy-Mechanismus, der einen Punkt (den Proxy) bei der Datenübertragung festlegt, ist aber nicht auf QoS ausgelegt, sondern auf die Überbrückung von Firewalls. Iteratives Proxying (ein „Proxy“ in jedem AS) ähnlich zu SOCKS wäre eine Möglichkeit, einen Pfad zu schalten; da SOCKS aber auf einen Proxy ausgelegt ist, wäre die Pfadschaltung sehr aufwändig.

In [BZKS 08] wird ein Zusammenhang zwischen der Durchsetzung neuer QoS-Technologien für Ende-zu-Ende-Verbindungen und neuen Geschäftsmodellen bzw. finanziellen Anreizen hergestellt.

Die Probleme der Ressourcenzuteilung, die im Zusammenhang mit SIP angesprochen wurden, werden im folgenden Abschnitt an einem Beispiel verdeutlicht.

Probleme von QoS im Internet am Beispiel ToS-Byte

1. **Konsistente Verfügbarkeit einer QoS-Technologie:** Die Interpretation des ToS-Byte im IPv4-Header ist nicht eindeutig, seine Semantik ist in drei RFCs definiert. Ein AS-Betreiber kann das ToS-Byte nach [Post 81], nach [NBBB 98], nach [RFB 01] interpretieren oder es ignorieren. Aber auch, wenn alle das ToS-Byte nach DiffServ interpretieren würden, wäre damit keine einheitliche Behandlung der Pakete über AS-Grenzen hinweg garantiert, da man mit DiffServ die Pakete zwar klassifiziert, aber keine konkreten Priorisierungsmechanismen vorschreibt.

2. **Zugriffskontrolle im eigenen Netz:** Das ToS-Byte muss an einer geeigneten Stelle gesetzt werden. Wenn jeder Knoten im IP-Netz das Feld setzen und verändern kann, bleibt es wirkungslos, da jeder seinen eigenen Paketen die höchste Priorität geben könnte. Die ISPs setzen das ToS-Byte daher an den Netz-Zugangsstellen normalerweise auf Null.

3. **Vertrauen und Kontrolle (inter-organisational, „Trust-Boundary“):** Die Zugriffskontrolle ist auf das eigene Netz beschränkt. An den Netzgrenzen, also auf den Border-Routern, muss definiert sein, wie die von außen eintreffenden Pakete behandelt werden. Entweder wird dem Nachbarn vertraut und das ToS-Byte unverändert weitergeleitet, oder es wird überschrieben, z. B. mit Null. Für QoS von Ende zu Ende müssten sich alle autonomen Systeme auf dem Weg vertrauen; setzt ein AS das ToS-Byte auf Null, so laufen die Maßnahmen der anderen ins Leere.

4. **QoS-Heuristik:** Der Vorschlag, das ToS-Byte anhand der verwendeten Ports zu setzen, um z. B. VoIP-Daten bevorzugt zu behandeln, führt bei Ende-zu-Ende-Verbindungen auch nicht zum Ziel. Einerseits ist es möglich, diese Ports für etwas Anderes zu verwenden, und andererseits ist diese Vorgehensweise nicht auf beliebige Anwendungen übertragbar bzw. konfigurierbar.

5. **Kooperationszwang:** Haben benachbarte Netzbetreiber kein Interesse an der Einführung einer QoS-Technologie wie DiffServ, oder kann man sich nicht über die einzusetzende Technologie und die entsprechenden Investitionskosten einigen, so sind die möglichen QoS-fähigen Pfade eingeschränkt, im schlechtesten Fall auf das eigene Netz.

Diese Probleme machen QoS-Garantien von Ende zu Ende unmöglich und verhindern neue Geschäftsmodelle („Verkauf“ von QoS-Parametern, insbesondere von Ende zu Ende). Das senkt auch die Bereitschaft der Netzbetreiber, in QoS-Technologie zu investieren.

Die Kombination von Best-Effort und Overprovisioning, die momentan weitgehend für die Qualität der IP-Übertragung im Internet verantwortlich ist, ist keine Lösung für diese Probleme. Es ist nicht möglich, mit Best-Effort und Overprovisioning einen Pfad zu schalten, der die Einhaltung spezifischer QoS-Parameter garantieren kann. Abgesehen davon ist im heutigen Internet ein echtes Overprovisioning nicht möglich, da eine TCP-Verbindung via Slow-Start ([Jaco 88] und [Brad 89]) in kürzester Zeit (den Download einer ausreichend großen Datei vorausgesetzt) jede Leitung ausschöpft und auf Paketverluste zur Stauvermeidung (Congestion Avoidance) angewiesen ist.

Das Beispiel Skype zeigt, dass Pfadschaltung im Internet funktionieren kann. Skype verwendet dazu allerdings ein Overlay-Netz und kann keine QoS-Garantien geben, sondern sucht nach Ressourcen und steuert den Datenverkehr der Verbindungen über das Overlay-Netz (siehe z. B. [BaSc 04]).

Diese Diplomarbeit stellt eine Lösung vor, wie Pfade einer spezifischen Dienstgüte auf IP-Ebene ohne Overlay geschaltet werden können. Die Lösung berücksichtigt insbesondere die genannten Probleme und funktioniert AS-übergreifend, ohne den Einsatz spezifischer QoS-Technologien vorauszusetzen.

4. Konzeption des WAN-Gateway

In diesem Kapitel wird das Gateway konzipiert, das den Pfadschaltungsdienst bereitstellt. Dafür wird das Gateway in funktionale Komponenten unterteilt, die alle auf einer physischen Komponente zusammengefasst und als eine Einheit in das Netz integriert werden können, die aber auch auf verschiedene physische Komponenten im Netz verteilt werden können. Die Unterteilung in funktionale Komponenten ist außerhalb des autonomen Systems transparent, dort bezeichnet WAN-Gateway bzw. Gateway die Vereinigung aller funktionalen Komponenten.

4.1. Funktionale Komponenten und Aufgaben

Abbildung 4.1 gibt einen Überblick über die funktionalen Komponenten sowie ihre Aufgaben und Interaktionen. Die funktionalen Komponenten sind beispielhaft auf physische Komponenten verteilt. Die funktionalen Komponenten sind:

- Der Pfadschaltungsdienst auf den Gateways
- Der Verbindungsauslösungsdienst auf einem Verbindungszugangspunkt
- Der Ressourcenverwaltungsdienst und die verwalteten Ressourcen
- Der Rewriting-Router und ein Rewriting-Dienst
- Ein Dienst zur Aktualisierung der Routing- und Topologieinformationen für den Pfadschaltungsdienst

Ein Mechanismus zur Bekanntmachung neuer Subnetze bzw. hier der QoS-Netze auf den Border-Routern ist in der Regel schon vorhanden, allerdings sind die Netze mit ihrer Bekanntmachung noch nicht als QoS-Netze identifiziert.

Die Aufgaben, die die funktionalen Komponenten erfüllen und die korrespondierenden Interaktionen sind (nummeriert mit Bezug zu Abbildung 4.1):

1. Kommunikation zur Pfadsuche und Pfadschaltung zwischen Verbindungsauslöser, Verbindungszugangspunkt und Pfadschaltungsdienst (Gateways)

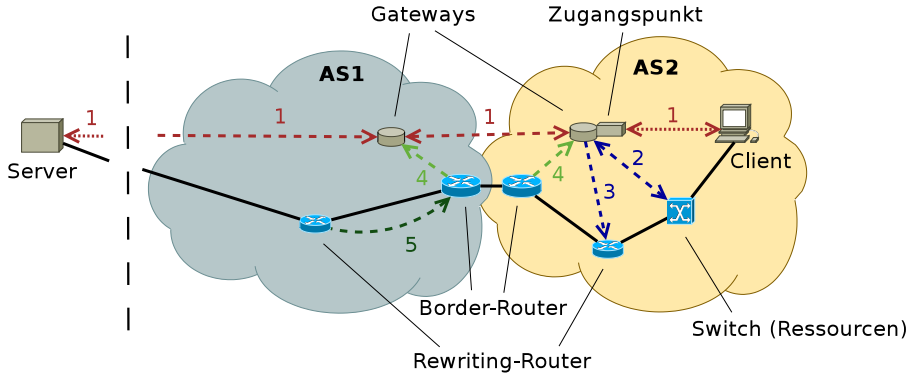


Abbildung 4.1.: Interaktionen des Pfadschaltungsdienstes auf den Gateways mit anderen (funktionalen) Komponenten

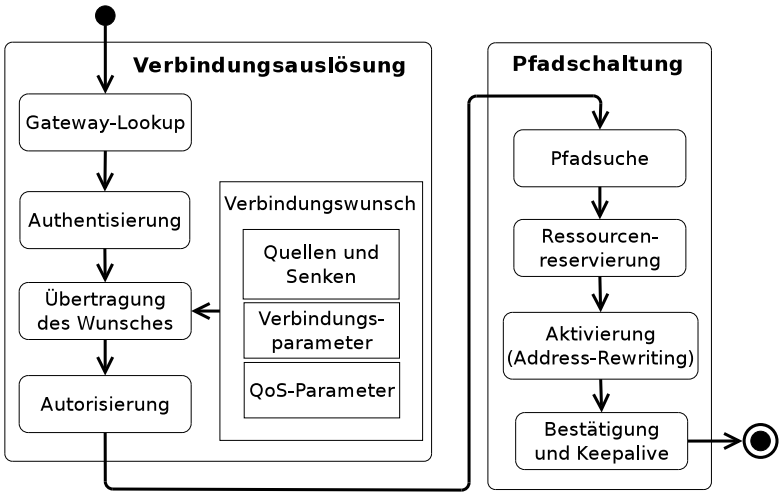


Abbildung 4.2.: Der Verbindungsaufbau unterteilt sich in Verbindungsauslösung und Pfadschaltung

2. Verwaltung der Ressourcen für die Einhaltung der QoS-Parameter
3. Adressierung der Verbindungspakete für das Routing in das nächste deterministisch erreichbare AS (*Address-Rewriting*), ausgelöst durch den Pfadschaltungsdienst
4. Update der Routing-Informationen durch die Border-Router (bzw. einen BGP-Speaker)
5. Bekanntmachung der eigenen QoS-Netze bei einem Border-Router (bzw. BGP-Speaker)

Diese funktionalen Komponenten können alle in das Gateway integriert werden, das über entsprechende Fähigkeiten (Zugriff auf Ressourcen, Routing für QoS-Adressen) verfügen muss. Oft ist es aber sinnvoll, einzelne oder alle funktionale Komponenten auszulagern, und über ein Backend mit dem Pfadschaltungsdienst zu verbinden.

In den folgenden Abschnitten werden die funktionalen Komponenten und ihr Zusammenspiel behandelt. Der Pfadschaltungsdienst und die Verbindungsauslösung werden in Abschnitt 4.2 konzipiert, die Ressourcenverwaltung in Abschnitt 4.4 und der Rewriting-Router und die Aktualisierung der Topologieinformationen in Abschnitt 4.5. Im Abschnitt 4.6 werden Fehler analysiert, die auf und zwischen den Komponenten auftreten können und Konzepte für deren Behandlung dargelegt und zum Abschluss des Kapitels werden in Abschnitt 4.7 Konzepte für Sicherheit und Abrechnung der Verbindungen skizziert.

4.2. Verbindungsmanagement

In Abschnitt 2.3 Anforderung 1 wird die konzeptionelle Aufteilung der Authentifizierung und Autorisierung zur Pfadschaltung in einen lokalen und einen globalen Teil gefordert, dementsprechend wird das Verbindungsmanagement in zwei Teile unterteilt.

Der lokale Teil besteht aus der *Auslösung, Nutzung und Beendigung* (und Vergütung) der Verbindung durch den Verbindungsnutzer bzw. Auslöser und der globale Teil behandelt die *Pfadschaltung und Pfadfreigabe* durch die Gateways. Der lokale Teil betrifft die Schnittstelle zwischen Verbindungsnutzer und Verbindungsanbieter, der globale Teil behandelt die Suche eines Pfades, die Reservierung von Ressourcen und die Determinisierung des IP-Routing entlang des Pfades sowie die Freigabe der Ressourcen und die Deaktivierung der Determinisierungsmaßnahmen.

Die Abbildungen 4.2 und 4.3 geben eine Übersicht über Verbindungsauf- und Abbau. Der Verbindungsauslöser muss zunächst ein passendes Gateway finden, das die Verbindung aufbauen kann (Gateway-Lookup). Bei diesem Gateway authentisiert er sich und überträgt den Verbindungswunsch.

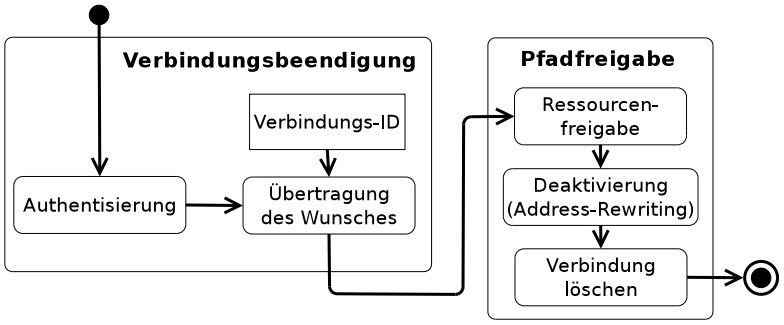


Abbildung 4.3.: Der Verbindungsabbau unterteilt sich in die Beendigung durch den Nutzer und die Pfadfreigabe

Das Gateway überprüft den Verbindungswunsch und autorisiert die Verbindung, falls der Auslöser dazu berechtigt ist.

Das Gateway initiiert die Pfadsuche, reserviert die Ressourcen und sorgt dafür, dass die Verbindungspakete dem gefundenen Pfad folgen, d. h. determinisiert das Routing durch die autonomen Systeme (durch Address-Rewriting). Der erfolgreiche Verbindungsaufbau wird dem Auslöser bestätigt und die Dauer der Verbindung wird signalisiert (Keepalive). Für den Abbau der Verbindung (Abbildung 4.3) authentisiert sich der Verbindungsauslöser wieder bei dem Gateway und identifiziert die zu löschende Verbindung. Das Gateway signalisiert die Freigabe der Ressourcen und deaktiviert die Determinisierung des Routings (d.h. das Address-Rewriting).

Der erste Teil kann szenariobedingt sehr unterschiedlich konzipiert werden. Auch für dasselbe Szenario kann jeder Verbindungsanbieter eine eigene Schnittstelle zu seinen Kunden definieren, daher ist der Abschnitt, der diesen Teil behandelt, weniger detailliert, aber breiter gefächert, als der Abschnitt über die Pfadschaltung.

Konzepte zur Auslösung von Verbindungsauf- und Abbau

Die in den Szenarien (Abschnitt 2.1) beschriebenen Mechanismen zur Verbindungsauslösung sind:

- Automatische Auslösung durch einen Dienst (Szenario Video-on-Demand und VoIP, Abschnitte 2.1.1 und 2.1.2)
- Auslösung durch Kommunikation mit dem Verbindungsanbieter (Szenario Videokonferenz in Abschnitt 2.1.3)

Die Auslösung „von Hand“ kann auf unterschiedliche Weise geschehen, etwa per Mail an den Anbieter, über ein Web-Interface oder anhand eines Clients. Bei all diesen Möglichkeiten (und genauso bei der automatischen Auslösung), wird der Verbindungswunsch irgendwann an ein Gateway übergeben. Der Übergang zu den Gateways muss vor unberechtigtem Zugriff geschützt werden.

Nach den Anforderungen aus Abschnitt 2.2.4 müssen sich die Gateways nur untereinander authentifizieren können, Verbindungsnutzer müssen daher, sobald ein Gateway die Pfadschaltung initiiert, bereits authentifiziert und zur Verbindungsauslösung autorisiert sein. Das Gateway, das die Pfadschaltung initiiert (das „Initiator-Gateway“) ist ab diesem Zeitpunkt „Verbindungsbesitzer“ und für die Verbindung (und den von ihr verursachten Aufwand) verantwortlich. Da die Verbindungsauslösung für die Gateways (abgesehen vom Initiator-Gateway) transparent ist und nur das Initiator-Gateway (und nicht der Auslöser) bei der Pfadschaltung bekannt sind, kann jedes Gateway die Auslösung beliebig implementieren.

Zwischen dem Initiator-Gateway und Verbindungsnutzer kann ein Verbindungsauslösungsdienst vermitteln, der vom Gateway authentifiziert werden kann und dem das Gateway vertraut. Der Verbindungsauslösungsdienst übernimmt die Autorisierung der Verbindungswünsche und die Authentifizierung der Verbindungsauslöser, hier müssen also die Daten über berechtigte Verbindungsauslöser, die Methoden zu deren Authentifizierung, und Informationen darüber, welche Art von Verbindungen bzw. wieviele Ressourcen der Auslöser verwenden darf, vorhanden sein.

Der Verbindungsauslösungsdienst ist in allen Szenarien einsetzbar: Wird der Verbindungswunsch per Mail übermittelt, so wird er von einem Mitarbeiter des Anbieters verarbeitet (d.h. authentifiziert, autorisiert und an den Dienst übergeben), das Web-Interface und der Client (der in irgendeiner Form auch bei der automatischen Auslösung zum Einsatz kommen muss) können direkt mit dem Dienst kommunizieren und müssen dort entsprechende Methoden und Daten zur Authentifizierung und Autorisierung hinterlegt haben.

Der Verbindungsauslöser ist nicht immer Verbindungsnutzer, daher muss er, neben Adressen für Quellen und Senken, auch eine „Kontakt- bzw. Management-Adresse“ angeben, an die die Verbindungsbestätigung, Verbindungsinformationen, Fehler, Verbindungsabbruch, Verbindungsende etc. gesendet werden. Im Szenario „Videokonferenz“ wäre die Management-Adresse die des Client, über den ein Mitarbeiter die Mailanfragen eingegeben hat. Nach erfolgreicher Pfadschaltung empfängt der Verbindungsauslöser die Verbindungsbestätigung und kann die Verbindung ab sofort nutzen (bzw. die Verbindungsdaten per Mail an den Verbindungsnutzer weitergeben).

Die Beendigung einer Verbindung funktioniert prinzipiell genauso, wie die Verbindungsauslösung: Der Verbindungsauslöser authentisiert sich bei dem Verbindungsauslösungsdienst und überträgt den Beendigungswunsch. Der Verbindungsauslösungsdienst leitet diesen an das Initiator-Gateway weiter, welches den Verbindungsabbau initiiert und bestätigt. Die Bestätigung ist nötig, da der Beendigungszeitpunkt möglicherweise abrechnungsrelevant ist. Der Verbindungsauslösungsdienst kann Teil der Gateways sein oder ausgelagert werden. Die konzeptionelle Trennung der Verbindungsauslösung macht es möglich, bei der Konzeption der Pfadschaltung die Verwaltung der Verbindungsnutzer und Auslöser auszublenden.

Zudem stellt sich die Frage, wo die Verbindung topologisch ausgelöst werden darf. Kanonische Kandidaten wären ein Gateway im AS des Auslösers oder ein Gateway im AS der Quelle bzw. eines Verbindungsendpunktes (siehe Abbildung 4.4). Wird der Pfad von einem Endpunkt-AS aus gesucht, vereinfacht das die Pfadsuche, allerdings müsste der Verbindungsauslösungsdienst eventuell Gateways in fremden (nicht benachbarten) autonomen Systemen kennen und diese müssten wiederum

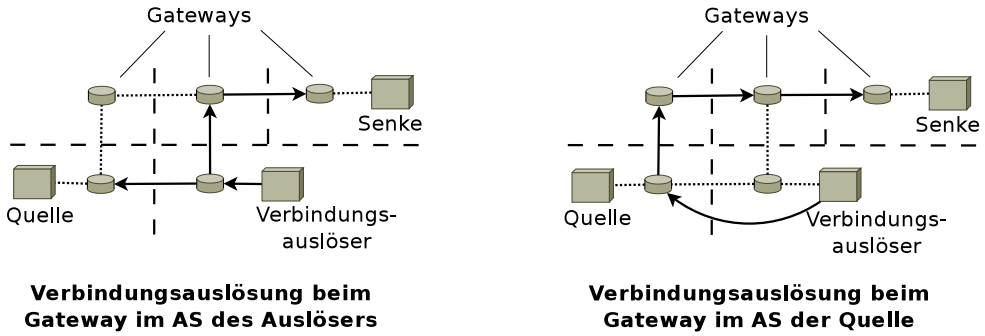


Abbildung 4.4.: Start der Pfadsuche und Wahl des Initiator-Gateways, links nach dem Ort des Auslösers, rechts nach dem Ort eines Verbindungsendpunktes

den Verbindungsauslösungsdienst authentifizieren können. Liegt das AS des Auslösungsdienstes auf einem (guten) Pfad zwischen den Endpunkten (wie in Abbildung 4.4 auf der linken Seite), wäre durch die Auslösung in diesem AS eine schnellere Pfadsuche möglich, da in beide Richtungen gleichzeitig gesucht wird.

Liegt das AS des Verbindungsauslösungsdienstes andererseits auf keinem oder einem sehr ungünstigen Pfad, so würde entweder der ungünstige Pfad gefunden, oder die Pfadsuche würde viel aufwändiger, als die Suche von einem Endpunkt aus: Um in diesem Fall trotzdem einen Pfad ohne Umwege zu bekommen, müssten entweder alle Gateways in beide Richtungen suchen, was bei komplexen Suchgraphen sehr aufwändig und fehleranfällig ist, oder den Suchgraph kennen und die Suche direkt an ein entferntes Gateway, das auf einem günstigen Pfad liegt, weitergeben.

Die Suchgraphen im Internet können, abhängig vom Ort und Abstand der Endpunkte, extrem komplex sein und der Suchgraph und seine Komplexität sind in der Regel unbekannt. Die Verwaltung fremder Verbindungsauslösungsdienste in Gateways macht die Pfadschaltung ebenfalls komplexer, daher wird, soweit nicht anders angegeben, bei der Konzeption der Pfadschaltung im nächsten Abschnitt davon ausgegangen, dass die Verbindungsauslösung immer im AS eines Endpunktes bzw. einer Quelle stattfindet und die Pfadsuche auch dort beginnt (d.h. bei einer unidirektionalen Unicast-Verbindung von Quelle nach Senke verläuft).

4.3. Konzeption der Pfadschaltung

Eine Verbindung hat auf jedem Gateway die in Abbildung 4.5 gezeigten Zustände. Die Verbindung kann in jedem Zustand abgebrochen werden, ein entsprechender Zustandsübergang in den Zustand `CLOSING` wurde der Übersichtlichkeit halber weggelassen. Die Definition eines globalen Verbindungszustands ist nicht offensichtlich. Eine Verbindung kann auf dem Initiator-Gateway bereits im Zustand `CLOSING` sein, während sie sich auf anderen Gateways noch im Zustand `SEARCHING` und `NEW` befinden kann. Da das Initiator-Gateway die Pfadschaltung in letzter Instanz steuert, kann man den Zustand der Verbindung dort am ehesten als den globalen Zustand der Verbindung bezeichnen.

Die Semantik der Zustände aus in Abbildung 4.5 ist folgende:

- `NEW`: Ein Verbindungswunsch ist eingetroffen und wird ausgewertet
- `SEARCHING`: Die Pfadsuche wurde angestoßen und es wird auf Ergebnisse gewartet
- `CONNECTING` Variante 1: Ein Pfad wurde ausgewählt und wird geschaltet
- `CONNECTING` Variante 2: Ein Pfad wird gesucht und „on-the-fly“ geschaltet
- `KEEPALIVE`: Der Pfad wurde geschaltet und kann genutzt werden
- `CLOSING`: Die Verbindung wurde beendet, es wird auf die Freigabe der Ressourcen gewartet

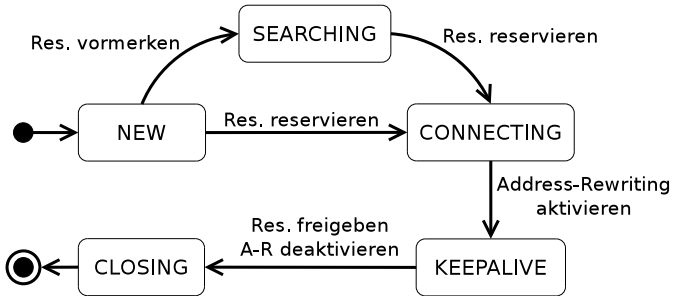


Abbildung 4.5.: Zustände einer Verbindung aus Gateway-Sicht

Nachdem ein Verbindungswunsch von dem Verbindungsauslösungsdienst akzeptiert und an den Pfadschaltungsdienst weitergeleitet wurde, beginnt die Pfadschaltung mit der Auswertung des Verbindungswunsches im Initiator-Gateway.

4.3.1. Auswertung des Verbindungswunsches (Zustand `NEW`)

1. Adressen der Endpunkte auswerten: Falls ein Endpunkt außerhalb des eigenen Zuständigkeitsbereichs liegt, Gateway(s) für den nächsten Pfadabschnitt suchen (Gateway-Lookup)
2. Verbindungsparameter auswerten (Unidirektionalität, Bidirektionalität etc.)
3. Verlauf der möglichen Pfade im eigenen AS bestimmen
4. Verfügbarkeit von Adressen prüfen, die für die Verbindung bzw. deren Routing über dieses Gateway geeignet sind
5. QoS-Parameter auswerten: QoS-Parameter an ein QoS-Backend übergeben (möglicherweise ist die Wahl des QoS-Backends vom Pfadverlauf abhängig) und die Erfüllbarkeit der QoS-Parameter prüfen lassen

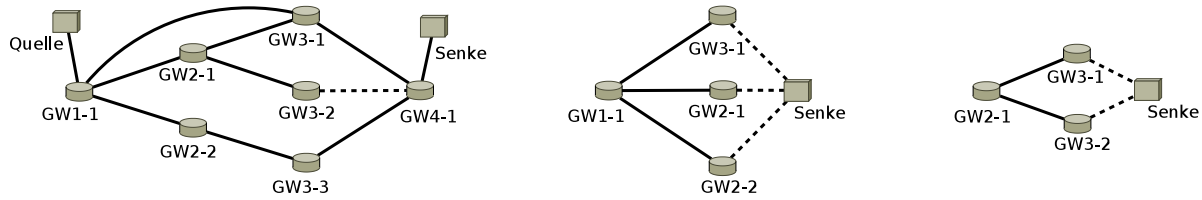
Die Auswertung eines Verbindungswunsches ist prinzipiell in jedem Gateway gleich. Der Verbindungswunsch wird abgelehnt, falls einer der folgenden Punkte *nicht* zutrifft:

- Ein Verbindungsendpunkt liegt im Einflussbereich des Gateways oder mindestens ein Gateway für den nächsten Pfadabschnitt wurde gefunden
- Adressen sind, soweit benötigt, verfügbar
- Das QoS-Backend hat die QoS-Parameter verstanden und hat die Erfüllbarkeit bestätigt

Falls auch keine anderen Gründe gegen die Bearbeitung des Verbindungswunsches vorliegen, wird er angenommen und mit der Pfadsuche begonnen und der Verbindungswunsch an ein oder mehrere gefundene Gateways weitergeleitet.

4.3.2. Pfadsuche (Zustand `SEARCHING`)

Es kann nicht davon ausgegangen werden, dass den Gateways ein vollständiger und aktueller Graph zur Verfügung steht, der alle Gateways miteinander verknüpft. Möglicherweise gibt es gar keinen eindeutigen Graphen: Je nach vertraglichen Abmachungen könnte ein Gateway eine Kante nur exklusiv für Verbindungen, die von einem bestimmten Gateway stammen, verwenden. Der Graph hinge in diesem Fall vom Verlauf der Suche ab, die diesen Graphen verwendet. Auch wenn für dieses Problem eine Lösung möglich wäre (z. B. die Verwendung mehrerer Graphen), machen lokale Besonderheiten dieser Art die Konzeption der Pfadsuche anhand von Graphen weniger attraktiv.



Globaler Suchgraph **Sicht von GW1-1** **Sicht von GW2-1**

Abbildung 4.6.: Globaler und lokaler Graph für die Pfadschaltungssuche, auf den gestrichelten Linien ist normales Routing, aber keine Pfadschaltung möglich (globale Sicht) bzw. die Möglichkeiten sind nicht sichtbar (lokale Sicht)

Für die Pfadschaltung ist die Kenntnis der Gateways in benachbarten autonomen Systemen unerlässlich, daher wird die Pfadsuche darauf aufbauend konzipiert, die Kenntnis entfernter Gateways wird nicht vorausgesetzt. Zusätzlich wird noch die Information benötigt, ob die Gateways in benachbarten autonomen Systemen auf einem möglichen Pfad vom aktuellen Gateway zur Senke liegen. Idealerweise werden die in Frage kommenden Gateways noch bewertet, sodass die besseren (die günstigeren, die mit höherer Erfolgsquote bei der Pfadsuche) zuerst für die Pfadsuche ausgewählt werden können. Ein *Gateway-Lookup* würde zu einer Zieladresse die in Frage kommenden Gateways in sortierter Form (die besten zuerst) ermitteln. Ist keine Bewertung der Gateways möglich, so sollte die Reihenfolge zufällig sein.

In Abbildung 4.6 ist ein globaler Suchgraph (der Graph, der alle möglichen Pfade enthält) abgebildet, sowie die zugehörigen Suchgraphen aus Sicht von GW1-1 und GW2-1. GW2-1 hat in diesem Beispiel keine Informationen darüber, dass zwischen GW3-2 und GW4-1 keine Pfadschaltung, sondern nur Routing ohne QoS möglich ist, d. h. der Graph von GW2-1 enthält unmögliche Pfade. Der globale Suchgraph zeigt, dass sich Pfade überschneiden können: GW3-1 ist sowohl direkt von GW1-1 aus erreichbar, als auch über GW2-1. Da GW1-1 und GW2-1 nur ihre eigenen Graphen kennen, können sie eine Überschneidung (z. B. durch Breitensuche) nicht voraussehen. Daher muss GW3-1 die Überschneidung erkennen (so können möglicherweise auch Zyklen erkannt werden). Das Erkennen einer Überschneidung führt bei GW3-1 dazu, dass die Suche nur einmal ausgeführt wird. Das Ergebnis könnte sowohl an GW2-1, als auch an GW1-1 gesendet werden. Damit würde die Wahl des Pfades „nach links“ verschoben. Alternativ kann GW3-1 auch einem der Gateways einen Korb geben und damit die Anzahl der Pfade, die GW1-1 zur Wahl hat, verringern.

Eine Überschneidung ist leicht zu erkennen, wenn jede Verbindung einen Identifikationsschlüssel bekommt. Adressen und Parameter eignen sich nicht als Schlüssel, da zwei Verbindungen mit denselben Parametern und Adressen sinnvoll möglich sind. Jede Verbindung soll daher eine eindeutige ID bekommen. Mit dieser ID sind Überschneidungen in der Pfadsuche erkennbar, Zyklen aber nicht. In Abwesenheit von Informationen über den Graphen wäre ein *Time-to-live-Mechanismus* eine Möglichkeit. Heuristiken, die mit der Zeit zwischen zwei Nachrichten arbeiten, sind nicht geeignet, da auch die Nachrichten von einfachen Überschneidungen in nahezu beliebigem Zeitabstand eintreffen können.

Beispiel: In Abbildung 4.6 könnte GW1-1 eine normale Breitensuche durchführen (d.h. gleichzeitig an GW2-1, GW2-2 und GW3-1 eine Nachricht senden) während GW2-1 zuerst einen Pfad über GW3-2 sucht. Nach einer Verzögerung stellt GW2-1 fest, dass kein Pfad über GW3-2 gefunden wurde und sendet eine Nachricht an GW3-1. Der Abstand zwischen dem Eintreffen der Nachricht von GW1-1 auf GW3-1 und dem Eintreffen der Nachricht von GW2-1 auf GW3-1 ist theoretisch beliebig, ein Zyklus kann auf diese Weise also nicht erkannt werden. Bei entsprechender Topologie ist auch ein mehrfaches korrektes Eintreffen solcher Nachrichten denkbar, das Zählen auf einem Gateway führt also auch nicht zum Ziel. Die später eintreffenden Nachrichten können zum Erfolg der Pfadsuche beitragen, die Verzögerung ist nicht unbedingt negativ, sie kann Teil der Pfadsuchstrategie auf den Gateways sein.

Alternativ zur Time-to-live könnte sich auch jedes Gateway in eine Liste der beteiligten Gateways eintragen, die mit der Nachricht mitgesendet würde. Ein Zyklus würde so, anders als durch Time-to-

live, sofort gefunden, allerdings würde die Nachricht dadurch einerseits immer größer und andererseits Informationen über den bisherigen Pfad enthalten, was möglicherweise unerwünscht ist (es könnte z. B. die Gleichbehandlung darunter leiden). Eine Möglichkeit, die Pfadsuche zeitlich und räumlich bzw. topologisch zu begrenzen, sollte sinnvollerweise auch zur Verfügung stehen. Für die topologische Begrenzung ist der Time-to-live-Mechanismus ebenfalls nutzbar, die zeitliche Begrenzung kann durch Timeouts erreicht werden.

Das Initiator-Gateway bestimmt ein globales Timeout, das evtl. auch durch den Verbindungsnutzer bei der Verbindungsauslösung beeinflusst werden kann. Die Timeouts in den Transit-Gateways sollten (solange keine Ressourcen belegt werden) relativ großzügig gewählt werden, damit das Initiator-Gateway die Suche nach Ablauf des globalen Timeouts abbrechen kann und die Suche nicht schon vorher abbricht. Sobald Ressourcen belegt werden, hängen die Timeouts auch wesentlich von geschäftlichen Interessen ab. Das globale Timeout ist dann weniger wichtig als der Preis für die Ressourcen.

Wurde ein Pfad gefunden und geschaltet, müssen eventuell noch laufende Suchen in anderen Zweigen abgebrochen werden. Dazu muss sich jedes Gateway merken, an welche Gateways die Suche weitergeleitet wurde. Ein Gateway an einem Überschneidungspunkt, das die Suche für zwei oder mehr Gateways durchführt (wie GW3-1 im oben gezeigten Beispiel), muss den Abbruch der Suche erst weiterleiten, wenn alle Gateways, die auf das Ergebnis warten, den Abbruch der Suche signalisiert haben (und nicht etwa bei der ersten Abbruchnachricht).

Manche QoS-Parameter, wie die Latenz, hängen stärker von der Länge des Pfades ab als andere (z. B. die Übertragungsrate). Es reicht also nicht, dass der Time-to-live-Mechanismus einfach Hops zählt, die Gateways müssen die von ihnen verursachte „Länge“ einschätzen. Das kann durch Anpassung der QoS-Parameter geschehen: Im Fall der maximal erlaubten Latenz würde jedes Gateway die Zeit, die die Pakete zum Durchqueren des eigenen autonomen Systems bräuchten, von dem maximal erlaubten Wert abziehen, bevor der Verbindungswunsch weitergeleitet würde. Die Veränderung der QoS-Parameter ist aber klompliziert (siehe die folgenden Absätze) und in einem Algorithmus für die Pfadsuche schwierig zu handhaben. Um die Kanten des Suchgraphen bewerten zu können, müsste der Suchalgorithmus jeden QoS-Parameter behandeln. Das ist komplex genug, wenn die Pfadschaltung auf vier QoS-Parameter mit eindeutig definierter Semantik beschränkt ist. Sobald neue QoS-Parameter dazukommen oder die definierte Semantik im aktuellen AS keine Entsprechung hat, ist die Pfadsuche mit einem algemeinen Algorithmus nicht mehr zu lösen.

Um allgemeine und vergleichsweise einfache Algorithmen konzipieren zu können, wird für die Veränderungen der Anforderungen an den Pfad, während der Pfadsuche, eine Metrik eingeführt. In diesem Abschnitt wird die Metrik für die Gewichtung der Kanten im Suchgraph verwendet und ihre Verwendbarkeit für die Pfadsuche analysiert. Wie die Metrik exakt definiert ist und wie sie berechnet wird, präzisiert der Abschnitt 4.4.3. Die Metrik hat vor dem ersten Kantenübergang einen Maximalwert (z. B. 100) und wird von jedem Gateway mindestens um den Teil verringert, den der Transport durch das eigene AS „kostet“. Unterschreitet die Metrik einen Schwellenwert (z. B. Null), so ist die Suche an dieser Stelle gescheitert. Sei beispielsweise eine maximale Latenz von 50 Millisekunden gefordert und für die Durchquerung des aktuellen autonomen Systems würden 5 Millisekunden benötigt, dann zieht das zuständige Gateway mindestens die „Latenz-Kosten“ von $(5/50) * 100 = 10$ von der Metrik ab. Sind mehrere QoS-Parameter betroffen, so wird mit jedem Parameter eine neue Metrik berechnet, das *Minimum der Ergebnisse ist die neue Metrik*. Der Vorteil dieser Vorgehensweise ist, dass ein einfaches Instrument zur Bewertung von Pfaden bzw. Kanten zur Verfügung steht und bei Einführung neuer QoS-Parameter nur eine Funktion zur Berechnung der Metrik bezüglich dieses Parameters implementiert werden muss.

Die Metrik ist insofern korrekt, dass jeder gefundene Pfad die QoS-Parameter erfüllt. Sie ist, wenn mehrere Parameter betroffen sind, unvollständig, da mit dieser Vorgehensweise nicht alle möglichen Pfade gefunden würden (siehe Abschnitt 4.4.3).

Von den in Abschnitt 2.3 Anforderung 2 geforderten QoS-Parametern sind Latenz, Jitter und Paketverlustrate betroffen, wobei sich nur die Latenz (bei Vernachlässigung von Jitter) linear berechnen lässt, für Jitter lässt sich linear nur der Worst-Case berechnen, d. h. wenn man davon ausgeht, dass ein Paket in jedem AS die maximale Zeit und das folgende Paket in jedem AS die minimale Zeit benötigt. Die Laufzeitdifferenz der beiden Pakete ist der Worst-Case-Jitter. Abhängig von der Topologie kann

dieser Fall aber extrem unwahrscheinlich, und der tatsächliche Jitter wesentlich geringer sein, sodass die lineare Berechnung bei der Pfadsuche keine guten Ergebnisse bringt. Da sich die Berechnung der Paketverlustrate auf Wahrscheinlichkeiten (z. B. die Wahrscheinlichkeit, dass ein Paket in einem Router verworfen wird) stützt und von vielen Parametern abhängt, ist sie ebensowenig linear, wie die Berechnung des Jitter.

Die Semantik der Metrik kann variiert werden. Sie kann die Erfüllbarkeit der QoS-Parameter binär bewerten (Parameter erfüllbar oder nicht) und die Rolle eines Hop-Limit übernehmen, wenn die Gateways zur Optimierung der Suche andere Methoden zur Anpassung der von der Art und Länge des Pfades betroffenen QoS-Parameter verwenden.

Ein Beispiel für die binäre Bewertung der QoS-Parameter wäre folgendes: Ein positiver Wert der Metrik bedeutet, dass die Parameter erfüllt werden können (und angepasst wurden) und das Hop-Limit noch nicht erreicht wurde, ein Wert kleiner oder gleich 0 bedeutet, dass sie nicht erfüllt werden können oder dass das Hop-Limit erreicht wurde.

Die Methoden (binär oder als Kantengewicht) könnten zwar auch gemischt werden, dadurch erhöht sich aber die Komplexität, wie das Beispiel „Latenz“ zeigt: Sei wieder 50 Millisekunden die maximal erlaubte Latenz und die Durchquerung der beiden ersten autonomen Systeme benötige jeweils 5 Millisekunden. Das 1. Gateway verringert die Metrik um 10 von 100 auf 90. Setzt das zweite Gateway den QoS-Parameter Latenz auf 45 und lässt die Metrik unverändert, so geht das nächste Gateway davon aus, dass 4,5 Millisekunden ($1/10 von 45$) verbraucht wurden und noch $9 * 4,5 = 40,5$ Millisekunden zur Verfügung stehen. Die Metrik müsste zur Vermeidung dieser Art von Fehlern also auch angepasst werden (in diesem Fall auf $100 - (100 * 1/9)$).

Die konsistente Vermischung beider Konzepte ist möglich und sogar transparent für die anderen Gateways, soll hier aber nicht weiter analysiert werden. Um zwischen binärer und nichtbinärer Verwendung der Metrik zu unterscheiden, kann für die binäre Verwendung ein anderer Wertebereich definiert werden, um eine unbeabsichtigte Vermischung beider Konzepte zu vermeiden. In dem Beispiel könnte das Hoplimit von 200 auf 101 heruntergezählt werden. Durch die Definition weiterer Wertebereiche kann die Semantik der Metrik nach Bedarf beliebig erweitert werden. Die Metrik kann auch zur *Steuerung* der Pfadsuche verwendet werden, indem sie auf „teuren“ Pfaden stärker heruntergesetzt wird, als es die QoS-Parameter erfordern. Die Semantik von „teuer“ ist dabei den Gateways überlassen. Unabhängig von den QoS-Parametern soll die Metrik in jedem Gateway um einen Mindestbetrag vermindert werden, um die Pfadlänge zu beschränken.

Die Metrik kommt auf den Gateways zweimal zum Einsatz: Einmal beim Eintreffen und Weiterleiten der Suche und einmal beim Eintreffen und Weiterleiten der Suchergebnisse. Trifft ein Verbindungswunsch ein, so bewertet die darin enthaltene Metrik den bereits zurückgelegten Abschnitt des Pfades. Das Gateway (bzw. das QoS-Backend im Gateway) bewertet den Verlauf des Pfades durch das eigene AS zu den nächsten Gateways und berechnet daraus, kombiniert mit der „eingetroffenen“ Metrik, eine neue Metrik für jedes Gateway, an das der Verbindungswunsch weitergeleitet wird. Unterschreitet die berechnete Metrik einen Schwellenwert (im Beispiel oben war das „Null“), so ist das entsprechende Gateway nicht in Reichweite und wird nicht in die Suche einbezogen. Sind alle Gateways außer Reichweite, so wird die Suche im aktuellen Gateway aufgegeben (nur die Suche, die von dem zur Metrik gehörenden Gateway ausgelöst wurde) und eine entsprechende Nachricht an das vorherige Gateway gesendet.

Sobald ein Gatway feststellt, dass der gesuchte Verbindungsendpunkt im eigenen Zuständigkeitsbereich liegt, wird, wie bisher, der Verlauf im eigenen AS bewertet (hier der Verlauf zum Endpunkt) und eine neue Metrik berechnet. Diese Metrik bewertet einen kompletten Pfad. Ist der Endpunkt in Reichweite (d.h. die Metrik unterschreitet nicht den Schwellenwert), so wird das vorherige Gateway über den gefundenen Pfad und die dazugehörige Metrik informiert, andernfalls wird dem vorherigen Gateway mitgeteilt, dass die Suche gescheitert ist.

Die Gateways, die den Verbindungswunsch weitergeleitet haben, warten danach auf ein Ergebnis der Pfadsuche in den nächsten Gateways. Sobald Ergebnisse eintreffen, kann das Gateway eine Auswahl treffen und eine Metrik an das vorherige Gateway weiterleiten. Nicht benötigte Teilpfade (Suchergebnisse) kann das Gateway entweder offenlassen, bis die Verbindung steht, oder vorher freigeben.

Wird kein Ergebnis gefunden oder dauert die Suche zu lange, so bricht das Gateway die Suche ab und sendet entsprechende Nachrichten an die Gateways, die noch suchen und die Gateways, die auf ein Ergebnis warten.

Das Abbrechen der Suche kann auch durch eine Aufforderung von den „linken“ Gateways erfolgen, in diesem Fall müssen die „rechten“ Gateways darüber benachrichtigt werden. Noch zu klären ist das Vorgehen bei einer Überschneidung der Suche. Treffen zwei Verbindungswünsche für dieselbe Verbindung ein, so kann das Gateway die Suche entweder zusammenführen, eine der beiden Metriken aussuchen und später die andere aus dem Suchergebnis berechnen, oder einen Verbindungswunsch zeitlich verschieben oder ablehnen. Ob das Zusammenführen sinnvoll ist, hängt davon ab, wie die Metrik berechnet wird. Wenn, wie im Beispiel Latenz, das Ergebnis der Pfadsuche einfach auf einen weiteren Vorgänger übertragen werden kann, so kann die Suche zusammengeführt werden (siehe auch das folgende Beispiel). Das Gateway hat auch die Möglichkeit, zu wählen, sowohl (bei Überschneidungen) den Vorgänger, als auch (bei mehreren Ergebnissen) den Nachfolger. Spätestens das Initiator-Gateway muss ein Ergebnis aussuchen.

Beispiel zur Pfadsuche

Die Pfadsuche wird hier nochmal an einem Beispiel erläutert (Abbildungen 4.7 und 4.8). Der Wertebereich für die Metrik ist wieder 0 – 100, bis auf GW2-1 und GW2-2 wenden alle Gateways Breitensuche an.

Pfadsuche in Abbildung 4.7

- GW1-1 ist das Initiator-Gateway, es sendet Verbindungswünsche mit Metrik 90 an GW2-1 und GW2-2 und einen Verbindungswunsch mit Metrik 70 an GW3-1
- GW2-1 berechnet eine „Länge“ von 5 in Richtung GW3-2 und eine Länge von 10 in Richtung von GW3-1 und entscheidet sich dafür, GW3-2 zuerst einen Verbindungswunsch zu senden. Nachdem GW3-2 keinen Pfad findet, sendet es einen Verbindungswunsch mit Metrik 80 zu GW3-1
- GW2-2 berechnet eine Länge von 60 im eigenen AS und sendet einen Verbindungswunsch mit Metrik 30 an GW3-3
- GW3-1 erhält zuerst einen Verbindungswunsch mit Metrik 70 von GW1-1 und sendet, nachdem eine Länge von 40 für den Pfadverlauf im eigenen AS berechnet wurde, zuerst einen Verbindungswunsch mit Metrik 30 an GW4-1; später trifft ein Verbindungswunsch mit Metrik 80 von GW2-1 ein und GW3-1 entscheidet sich dafür, einen weiteren Verbindungswunsch (mit Metrik 40) an GW4-1 zu senden, da mit dieser Metrik möglicherweise mehr Pfade gefunden werden können
- GW3-2 erhält einen Verbindungswunsch von GW3-1, hat aber keine Möglichkeiten, einen Pfad in Richtung Senke zu schalten und lehnt daher die Suche ab
- GW3-3 erhält einen Verbindungswunsch von GW2-2 mit Metrik 30, berechnet eine Länge von 20 für das eigene AS und sendet einen Verbindungswunsch mit Metrik 10 an GW4-1
- GW4-1 erhält drei Verbindungswünsche, jeweils mit den Metriken 10, 30 und 40; der Verbindungswunsch mit der Metrik 40 wird als Update für den Verbindungswunsch mit der Metrik 30 gewertet, da er von demselben Gateway stammt; die Länge des Pfadverlaufs bis zur Senke beträgt 20, d. h. es wurde ein Pfad mit der Metrik 20 (aus Sicht von GW4-1) bzw. zwei Pfade mit den Metriken 10 und 20 (aus Sicht von GW3-1) gefunden

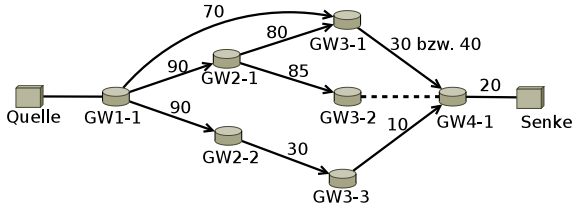


Abbildung 4.7.: Berechnung der Metriken und Weiterleitung des Verbindungswunsches von links nach rechts

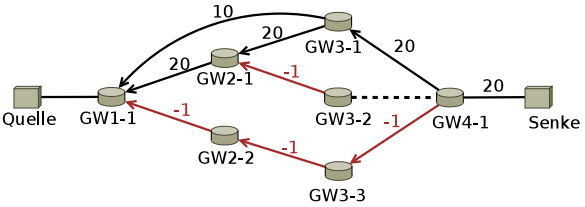


Abbildung 4.8.: Verarbeitung und Weiterleitung der Metriken von rechts nach links

Auswertung der Pfadsuche in Abbildung 4.8

- GW4-1 sendet eine Nachricht mit einer Pfadbestätigung (Metrik 20) an GW3-1 und benachrichtigt GW3-3 über die fehlgeschlagene Suche
- GW3-3 leitet die Nachricht weiter an GW2-2
- GW3-2 benachrichtigt GW2-1 über die fehlgeschlagene Suche
- GW3-1 berechnet aus der erhaltenen Metrik eine Pfadbestätigung (Metrik 10) für GW1-1 und leitet die Pfadbestätigung mit Metrik 20 an GW2-1 weiter
- GW2-2 benachrichtigt GW1-1 über die fehlgeschlagene Suche
- GW2-1 erhält eine Nachricht über die gescheiterte Pfadsuche von GW3-2 und wartet aber auf die Nachricht von GW3-1; sobald die Pfadbestätigung (Metrik 20) von GW3-1 eintrifft, wird sie an GW1-1 weitergeleitet
- GW1-1 erhält eine Nachricht über die fehlgeschlagene Suche von GW2-2 und eine Pfadbestätigung (Metrik 10) von GW3-1; GW1-1 kann entweder auf GW2-1 warten oder den Pfad über GW3-1 wählen und die Suche bei GW2-1 abbrechen

In diesem Beispiel wurde *Breitensuche* verwendet, das kann zur Folge haben, dass in manchen Gateways viele Suchen gleichzeitig bearbeitet werden, von denen ein großer Teil, bedingt durch die Breitensuche, abgebrochen wird. Hier stellt sich die Frage, wann und wie die Ressourcen für die Pfadschaltung reserviert werden. Wenn sie bereits bei der Suche exklusiv (d.h. ohne Überbuchung) reserviert werden, führt das auf den Gateways schnell dazu, dass alle Ressourcen für Pfadsuchen belegt sind, die mit hoher Wahrscheinlichkeit an dieser Stelle abgebrochen werden. Für den Einsatz bei der Breitensuche ist eine exklusive Reservierung also nicht geeignet, hier müssen andere Konzepte gefunden werden (siehe Abschnitt 4.4).

Ist bei der Pfadsuche die Auswahl des nächsten Gateways einfach, weil nur wenige Gateways in Frage kommen oder weil bekannt ist, welche Gateways die besten Kandidaten für die Pfadsuche sind, so kann auch *Tiefensuche* sinnvoll sein. Bei der Tiefensuche sind weniger Gateways gleichzeitig mit der Suche beschäftigt und alle suchenden Gateways sind, sobald ein Pfad gefunden wurde, an der Pfadschaltung beteiligt. Gateways, die nicht an der Pfadschaltung beteiligt werden können, erkennen das, bevor ein Pfad gefunden wurde. Hier ist das Risiko geringer, dass exklusiv reservierte Ressourcen nicht zum Einsatz kommen, daher könnte man Tiefensuche und exklusive Reservierung von Ressourcen sinnvoll kombinieren.

Ohne exklusive Reservierung von Ressourcen ist die Tiefensuche dagegen weniger sinnvoll, da die Suche dann ebenfalls abgebrochen werden kann, obwohl ein Pfad gefunden wurde. Der Erfolg der Tiefensuche mit exklusiver Reservierung hängt nur vom nächsten, nicht mehr vom vorherigen Gateway ab. Würde beispielsweise GW2-1 in Abblidung 4.8 die Suche kurz vor ihrem Ende abbrechen, weil die lokalen Ressourcen anderweitig vergeben wurden, so muss ein neuer Pfad gesucht werden, der bei Verwendung von Breitensuche möglicherweise schon bekannt und geschaltet wäre.

Die Konzeption der Pfadsuche ist hier in Stichpunkten noch einmal zusammengefasst:

- Initiator-Gateway als Initiator der Pfadsuche
- Suche beschränkt auf das bzw. die nächsten Gateways (Gateway-Lookup)
- Weiterleitung der Suche an das bzw. die nächsten Gateways
- Time-to-live-Mechanismus für die Verbindungssuche
- Timeouts für Wartezeiten auf den Gateways
- eindeutige Verbindungs-ID
- Überschneidungserkennung und -berücksichtigung (Pfadsuche und Abbruch der Suche)
- Zustandsspeicher für jeden an der Pfadsuche beteiligten Kommunikationspartner (z. B. um einen Abbruchsnachricht an alle zu schicken, die noch einen Pfad suchen)
- Metrik mit global bekanntem Wertebereich zur Bewertung von Pfaden und Pfadabschnitten, impliziert Time-to-live-Mechanismus bzw. Hop-Limit
- Semantik der Metrik als reines Hop-Limit durch anderen Wertebereich
- Exlusiv reservierte Ressourcen bei der Suche implizieren sinnvolle Tiefensuche
- Nicht exklusiv reservierte Ressourcen ermöglichen Breitensuche, Ressourcen werden „vorgemerkt“ (siehe Abschnitt 4.4)

War die Pfadsuche erfolgreich, so folgt die Schaltung bzw. Signalisierung des Pfades, die im folgenden Abschnitt behandelt wird.

4.3.3. Pfadschaltung (Zustand `CONNECTING`)

Die Pfadschaltung erfolgt in zwei Schritten:

1. Beginnend mit dem Initiator-Gateway werden der Verbindung in jedem Gateway auf dem Pfad Ressourcen und eine Rewriting-Adresse zugeordnet und die Aktivierung des Pfades dem jeweils nächsten Gateway signalisiert; ist die Verbindung bidirektional, so wird zusammen mit der Signalisierung eine Adresse für die Determinisierung des Routing an das nächste Gateway geschickt (siehe Abschnitt 4.5)
2. Die Aktivierung des Pfades wird, ausgehend vom Gateway im AS der Senke, dem vorherigen Gateway bestätigt und eine Adresse für die Determinisierung des Routing an das vorherige Gateway gesendet; mit dem Versenden der Bestätigung wird die Determinisierung des Routing (das Address-Rewriting) aktiviert (siehe Abschnitt 4.5); im Initiator-Gateway wird der Verbindungsauslöser benachrichtigt und erhält eine Verbindungsadresse

Der Signalisierung des Pfades kann eine Pfadsuche vorausgehen. Ist das der Fall, so wird vom Initiator-Gateway ein Gateway ausgesucht, das einen oder mehrere Pfade gefunden hat. Diesem wird die Pfadschaltung signalisiert. Falls mehrere Pfade gefunden wurden, wird ein Pfad anhand der Metrik ausgesucht, welcher zusammen mit der Signalisierung an das nächste Gateway gesendet wird.

Eine Differenzierung der Pfade über die Metrik hinaus ist nicht vorgesehen, dafür wären topologische Informationen nötig. Wenn ein Gateway zwei Pfade oder Pfadabschnitte mit derselben Metrik findet, kann es lokal (d.h. für das vorhergehende Gateway transparent) entscheiden, welcher Pfad bei der Signalisierung verwendet wird. Wenn bei der Pfadsuche keine Ressourcen (exklusiv) reserviert wurden,

kann die Pfadschaltung in einem Gateway an der Zuordnung der Ressourcen scheitern. In diesem Fall muss das Scheitern dem vorherigen und dem nachfolgenden Gateway mitgeteilt werden. Falls auf dem vorherigen Gateway ein weiterer Pfad gefunden und das Ergebnis gespeichert wurde, kann dieser für die Signalisierung verwendet werden, andernfalls muss es entweder einen neuen Pfad suchen oder das Scheitern der Signalisierung weiterleiten.

Auf Gateways, auf denen die Signalisierung gescheitert ist, kann die Verbindung nach der Benachrichtigung der benachbarten Gateways gelöscht werden, d.h. vorgemerkte bzw. reservierte Ressourcen freigegeben, aktives Rewriting deaktiviert und interne Verbindungsdaten gelöscht werden. Sobald die Signalisierung das Gateway im AS der Senke erreicht hat, sind alle Adressen und Ressourcen reserviert, die Verbindung kann ab jetzt nur noch durch Fehler (Ausfälle, Abstürze etc.) scheitern. Das Gateway im AS der Senke (das „letzte Gateway“) bestätigt dem vorherigen Gateway die erfolgreiche Signalisierung, sendet ihm eine Rewriting-Adresse und aktiviert das Address-Rewriting. Dieser Vorgang wiederholt sich in jedem Gateway bis zum Initiator-Gateway. Dort wird ebenfalls das Address-Rewriting aktiviert, die Rewriting-Adresse übernimmt hier aber die Rolle einer Verbindungsadresse für den Verbindungsnutzer.

Geht der Signalisierung des Pfades keine Pfadsuche voraus, so wird das nächste Gateway nicht anhand der Suchergebnisse ausgesucht, sondern genauso wie in der Pfadsuche. Der Ablauf von „links nach rechts“ ist derselbe wie bei der Tiefensuche mit exklusiv reservierten Ressourcen, der Ablauf von „rechts nach links“ entspricht dem der Pfadschaltung. Der einzige Unterschied zur oben beschriebenen Tiefensuche ist der, dass nach der Suche kein Pfad zur Signalisierung angeboten wird, sondern in einem Ablauf die fertige Verbindung aufgebaut wird.

Nachdem der Verbindungsauslöser und der Verbindungsnutzer über die erfolgreiche Pfadschaltung informiert wurden, kann die Verbindung genutzt werden. Um Ressourcenknappheit oder Ausfälle direkt (und nicht über eine Beschwerde des Verbindungsnutzers) zu bemerken, wird der eigene Pfadabschnitt von jedem Gateway überwacht und die Nachbar-Gateways darüber informiert. Die Verbindung befindet sich jetzt im Zustand `KEEPALIVE`. Die Benachrichtigung der Nachbarn über den Zustand der Verbindung ist sinnvoll, da auf diese Weise verwaiste Verbindungen verhindert werden können. Andernfalls würde eine Verbindung bzw. ein Teil von ihr weiter bestehen, wenn ein Gateway ausfällt: Da immer nur das nächste und das vorherige Gateway bekannt sind, würde bei einem Ausfall keine Kommunikation zwischen dem vorigen und dem nächsten Gateway möglich sein (siehe auch Abschnitt 4.6).

4.3.4. Verbindungsabbau (Zustand `CLOSING`)

Wenn der Verbindungsnutzer die Verbindung beenden möchte, übergibt er den Auftrag dazu an den Verbindungsauslösungsdienst. Dieser löst den Verbindungsabbau auf dem Initiator-Gateways aus. Die Gateways, angefangen beim Initiator-Gateway, geben die von der Verbindung belegten Ressourcen frei, deaktivieren das Address-Rewriting und lösen im nächsten Gateway den Verbindungsabbau aus. Sobald der Verbindungsabbau vom nächsten und dem vorherigen Gateway (bzw. dem Verbindungsauslösungsdienst) bestätigt wurde, können die internen Verbindungsdaten gelöscht werden und die Verbindung gilt auf diesem Gateway als beendet.

4.4. Ressourcenverwaltung und QoS-Parameter

Ausgehend von den vorhanden QoS-Technologien wird in Abschnitt 4.4.1 zuerst die Ressourcenverwaltung über QoS-Backends konzipiert. Im darauf folgenden Abschnitt 4.4.2 werden die QoS-Parameter ausgewählt, die direkt in das Pfadschaltungsprotokoll integriert werden sollen, und deren Semantik und die Syntax definiert. Der Abschnitt 4.4.3 widmet sich der Definition der bei der Pfadsuche verwendeten Metrik und ihrer Semantik in Abhängigkeit von den QoS-Parametern und gibt an, wie sie berechnet wird.

4.4.1. QoS-Backends

Wie in Kapitel 3 an Beispielen dargelegt wurde, gibt es in IP-Netzen zwei grundlegende QoS-Methoden. Einerseits können Ressourcen einer Verbindung eindeutig und exklusiv zugeordnet werden. Diese Ressourcen sind nicht von der Auslastung der Router abhängig, sondern stehen der Verbindung nach der Reservierung garantiert zur Verfügung.

Andererseits können die Pakete einer Verbindung einer bestimmten Verkehrsklasse zugeordnet werden. Die Ressourcen werden dann unter den verschiedenen Verkehrsklassen aufgeteilt. Die Zuteilung der Ressourcen zu einer Verbindung hängt von der Auslastung, von den Scheduling-Algorithmen und vom Zufall ab. Garantien können hier nicht gegeben werden, allerdings ist die Umsetzung vergleichsweise einfach.

Die IP-Vertreter der beiden Methoden sind Intserv/RSVP und DiffServ. In Kapitel 2 Abschnitt 2.2.1 wurde die Integration bestehender Infrastruktur gefordert, daher sollen beide Methoden in die Pfadschaltung integriert werden. Um QoS ohne Garantien nicht auf Best-Effort-Routing mit Prioritäten zu beschränken, kann die Verbindung bzw. die Auslastung der Ressourcen im eigenen AS überwacht, und bei Nichteinhaltung der QoS-Parameter reagiert werden, sodass QoS über Garantien oder über Priorisierung und Überwachung erreicht wird.

Um den unterschiedlichen Herangehensweisen in der Pfadschaltung Rechnung zu tragen, wird eine Option eingeführt, die Garantien (wie bei IntServ) für die Verbindung fordert, sodass trotz der Abstraktion der QoS-Technologie eine gewisse Konsistenz in der Umsetzung der QoS-Parameter gewährleistet ist.

Verschiedene Technologien für die Umsetzung der QoS-Parameter wurden in Kapitel 3 angesprochen, auf Schicht 3 des OSI-Schichtenmodells sind das IntServ/RSVP und DiffServ, unterhalb von Schicht 3 sind MPLS (mit RSVP-TE) und ATM verbreitet.

Die QoS-Backends benötigen für die Reservierung von Ressourcen die IP-Adressen der Verbindungspakete und sind möglicherweise an der lokalen Determinisierung des Routing beteiligt. Da die Adressen bei der Pfadsuche noch nicht feststehen, muss die Reservierung von Ressourcen in zwei Schritten geschehen.

1. Vormerken der Ressourcen bei der Pfadsuche
2. Reservieren der Ressourcen während der Pfadschaltung, sobald die IP-Adressen und der Pfadverlauf feststehen

Das Vormerken der Ressourcen kann, je nach Suchalgorithmus, die Ressourcen überbuchen, je wahrscheinlicher es ist, dass eine Verbindung nicht zustande kommt, umso stärker kann überbucht werden.

Während der Pfadschaltung ist die Semantik der Reservierung nicht ganz konsistent: Damit die Pfadschaltung nicht mehr „von links“ aufgrund von Ressourcenengpässen scheitern kann, sollen die Ressourcen reserviert werden, sobald die Aufforderung zur Pfadschaltung eintrifft. Zu diesem Zeitpunkt wird das nächste Gateway ausgewählt, der Pfad steht also schon (fast) fest. Die Adresse, die vom nächsten Gateway vergeben wird, ist aber noch nicht bekannt, d. h. bei Backends, die die Adresse zur Reservierung der Ressourcen auf den Netzkomponenten benötigen, müssen die Ressourcen für die Verbindung solange auf andere Weise blockiert werden, bis die Adresse bekannt ist. Solange die Ressourcen zentral verwaltet werden, ist das kein Problem, sobald mehrere Gateways auf dieselben Ressourcen zugreifen, ist Synchronisation notwendig. Hier stellt sich die Frage, ob und wie ein AS seine Ressourcen unter Gateways und QoS-Backends aufteilt. Es wäre möglich, von einem Gateway aus, mehrere QoS-Backends anzusprechen, die wiederum jeweils für einen Abschnitt zuständig sind. Alternativ könnte einem QoS-Backend ein Gateway zugeordnet werden (siehe Abbildung 4.9).

Zur Lösung dieses Problems können die QoS-Backends bei der Ressourcenreservierung einen Zwischenschritt einfügen. Bevor die Ressourcen auf den Netzkomponenten reserviert werden, werden sie vom QoS-Backend virtuell reserviert. Das QoS-Backend führt Buch über die Reservierungen und verhindert eine Überbuchung durch Reservierungen. Die Zuordnung von QoS-Backends zu Pfadabschnitten muss dafür eindeutig sein, mehrere unabhängige bzw. nicht synchronisierte Instanzen zur

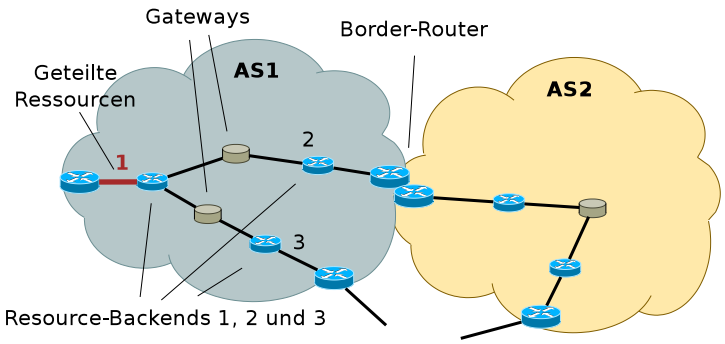


Abbildung 4.9.: Die Verfügbarkeit und Reservierung von Ressourcen hängt vom Verlauf des Pfades im eigenen AS ab

Ressourcenverwaltung eines Abschnitts führen zu Inkonsistenzen.

Da die Informationen zur Verfügbarkeit von Ressourcen bei den QoS-Backends liegen, liegt es nahe, auch die Entscheidung über das Vormerken von Ressourcen dort zu fällen. Um auch die Auswahl des geeigneten QoS-Backends, und damit die gesamte Ressourcenverwaltung, für den Pfadschaltungsdienst transparent zu machen, übernimmt auf jedem Gateway ein Ressourcenverwaltungsdienst die Aufgabe, die QoS-Parameter und die Adressen (bei der Suche die Adressen der Gateways, bei der Pfadschaltung die Rewriting-Adressen) auszuwerten.

Zuerst wird ausgewertet, welche QoS-Backends an der Verbindung bzw. am Routing der übergebenen Adressen beteiligt sind. Diesen Backends werden dann die QoS-Parameter für die Reservierung bzw. das Vormerken übergeben. Sie bestätigen den Vorgang oder lehnen ihn ab. Der Ressourcenverwaltungsdienst prüft die Auswertung auf Konsistenz. Hat ein QoS-Backend die Reservierung bzw. Vormerkung abgelehnt, so müssen die Ressourcen auf den anderen Gateways sofort wieder freigegeben werden. Hat kein Backend abgelehnt, so kann der Ressourcenverwaltungsdienst eine Metrik berechnen und dem Pfadschaltungsdienst übergeben.

- Ressourcenverwaltung in zwei Schritten: Vormerken der Ressourcen bei der Pfadsuche und Reservieren der Ressourcen während der Pfadschaltung
- Abstraktion durch QoS-Backends
- Ein QoS-Backend für jeden topologisch relevanten Abschnitt, hier findet die „Buchführung“ und die Kommunikation mit den Netzkomponenten statt
- Ein Ressourcenverwaltungsdienst pro Gateway, der die QoS-Anfragen an die richtigen QoS-Backends weiterleitet und aus den Antworten eine Metrik berechnet
- Mögliche QoS-Backends für IntServ/RSVP, MPLS, ATM, DiffServ

4.4.2. Austausch und Auswertung von QoS-Parametern

In Kapitel 2 ergab die Analyse der Szenarien, dass vier QoS-Parameter direkt in das Pfadschaltungsprotokoll integriert werden sollen:

- Minimale Übertragungsrate
- Maximale Latenz
- Maximaler Jitter
- Maximale Paketverlustrate

Die Syntax und Semantik dieser Parameter müssen noch definiert bzw. präzisiert werden. Die Parameter werden als Wert bzw. Zahl mit feststehender Einheit dargestellt. Als Einheiten werden *kBit/s*,

Millisekunden (Latenz und Jitter) und *die Anzahl verlorener Pakete pro* 10^9 *gesendeter Pakete* definiert. Die Werte beziehen sich auf den vollständigen Pfad von Quelle zu Senke. Bei einer bidirektionalen Verbindung gelten die Parameter in beide Richtungen. Asymmetrische Verbindungen sind mit dieser Definition der QoS-Parameter nicht möglich. Dafür können entweder zwei unidirektionale Verbindungen verwendet, oder neue, von der Richtung abhängige QoS-Parameter definiert werden.

Die Definitionen der vier QoS-Parameter bedürfen noch einer Präzisierung: Die Übertragungsrate wird in Zeitabständen gemessen, die kleiner sind als die maximale Latenz (falls eine maximale Latenz gefordert wurde, andernfalls wird pro Sekunde gemessen) und in kBit/s angegeben. Die maximale Latenz einer Verbindung definiert die Obergrenze für die Zeit, die zwischen Senden und Empfangen eines zu dieser Verbindung gehörenden Pakets (Ende zu Ende) verstreichen darf. Jitter definiert die maximal zulässige Latenz-Differenz von zwei aufeinander folgenden, übertragenen Paketen. Alternative Definitionen, wie z. B. die in [DeCh 02] (Packet-Delay-Variation, RFC 3393), müssen zusätzlich definiert und eingebettet werden (siehe unten). Die Paketverlustrate spezifiziert die Anzahl der Pakete, die verloren gehen oder zu spät eintreffen (falls eine maximale Latenz gefordert wurde), bezogen auf 10^9 gesendete Pakete[1].

In Abschnitt 4.4.1 wurde auf prinzipielle Unterschiede in der Umsetzung von QoS hingewiesen, manche Backends sind in der Lage, Ressourcen exklusiv zu reservieren und deren Verfügbarkeit zu garantieren, andere Backends priorisieren den Datenverkehr einer Verbindung nur und überwachen die Einhaltung der QoS-Parameter. Um diese prinzipiellen Unterschiede zu berücksichtigen, wird für die Semantik der vier QoS-Parameter die Option *„garantierte Ressourcen“* eingeführt.

Für eine Klasse von Verbindungen, die oft und regelmäßig geschaltet werden, könnten QoS-Anforderungen anhand einer ID identifiziert werden, die es ermöglicht, der Verbindung Ressourcen zuzuordnen, ohne die QoS-Parameter auszuwerten und Kapazitäten zu überprüfen. Dies soll anhand eines Beispiels verdeutlicht werden: Die Verbindung mit der QoS-Parameter-ID X bekommt Leitung Y von der Art Z, da Leitungen der Art Z für diese Anwendung erfolgreich getestet wurden. Die Parameter einer Z-Leitung spielen bei der Schaltung des Pfades keine Rolle, sie muss nur identifiziert werden können. Auf diese Weise können etablierte Zuordnungen von Ressourcen zu QoS-Anforderungen in die Pfadschaltung integriert werden. Die Syntax einer QoS-ID kann im Protokoll definiert werden, die Semantik muss unter den beteiligten Gateways vereinbart werden.

Neben den vier verbreiteten QoS-Parametern gibt es viele weitere sinnvolle Möglichkeiten, QoS-Parameter zu definieren. Ein Beispiel aus dem Videokonferenz-Szenario wäre die Verfügbarkeit der Verbindung für eine bestimmte Dauer. Um das Pfadschaltungsprotokoll nicht auf die integrierten Parameter einzuschränken, soll ein Formalismus zur Übertragung zusätzlicher QoS-Parameter konzipiert werden. Im Unterschied zur QoS-ID, die Verbindungen und Ressourcen klassifiziert, ist es hier möglich, den Parameter individuell zu kombinieren und zu bemessen. Auf diese Weise kann z. B. die Dauer der Verfügbarkeit einer Verbindung genau angegeben werden, eine ID dafür zu definieren, ist in diesem Fall nicht sinnvoll. Mit der Einbettung neuer QoS-Parameter können auch Informationen für die Pfadsuche definiert oder die Umsetzung der QoS-Parameter präzisiert werden.

Die QoS-Parameter, die von einer Verbindung eingehalten werden sollen, werden bei der Pfadsuche und der Signalisierung des Pfades übertragen. Sie werden nicht direkt vom Pfadschaltungsdienst durchgesetzt, sondern von einem QoS-Backend, das sie für die lokal verfügbare QoS-Technologie übersetzt. Das Backend überprüft bei der Pfadsuche, ob die QoS-Parameter eingehalten werden können und sorgt nach der Signalisierung des Pfades für deren Einhaltung.

Die Verwendbarkeit eingebetteter, nicht vom Pfadschaltungsprotokoll definierter QoS-Parameter hängt also vom QoS-Backend ab. Das Pfadschaltungsprotokoll könnte zwar beliebige QoS-Parameter übertragen, ein Pfad kann aber nur gefunden werden, wenn die anderen Gateways diese QoS-Parameter verstehen. Die Betreiber der Gateways müssen also vorher die Definition dieser QoS-Parameter austauschen und festlegen, wie sie umgesetzt werden können. Da die verwendbaren QoS-Parameter von den eingesetzten QoS-Backends abhängen, soll die Definition (Syntax und Semantik) von neuen QoS-Parametern sowie deren Bekanntmachung an die QoS-Backends delegiert werden.

[1] 10^9 ist menschenlesbar und durch 32 Bit darstellbar.

Die Gateways bzw. das Pfadschaltungsprotokoll übertragen also die QoS-Parameter und übergeben sie zur Auswertung an das Backend. Das Pfadschaltungsprotokoll muss nur die Antwort des QoS-Backends auswerten können, nicht die Parameter. Wird die Metrik zur Pfadsuche verwendet, kann vom QoS-Backend einfach eine Metrik berechnet werden. Wird die Metrik nicht verwendet, so ist die Schnittstelle zum QoS-Backend wesentlich komplexer: Das QoS-Backend muss dem Pfadschaltungsdienst nicht nur die Bewertung der QoS-Möglichkeiten an den Pfadschaltungsdienst übergeben, sondern auch die QoS-Parameter anpassen, bevor sie an das nächste Gateway weitergeleitet werden.

Das folgende Beispiel verdeutlicht das Problem: Sei ein eigener QoS-Parameter für die Latenz definiert, der Informationen darüber enthält, welcher Anteil der erlaubten Latenz schon „verbraucht“ ist (z. B. „maximale Latenz = 50 ms, davon sind 30 ms bereits verbraucht“). Kommen zwei Gateways für die Weiterleitung der Verbindungssuche in Frage, von denen eines in 10 und das andere in 20 Millisekunden erreichbar ist, so muss das QoS-Backend einerseits den QoS-Parameter in Abhängigkeit vom nächsten Gateway ändern und andererseits dem Gateway ein Entscheidungskriterium geben, welches der nächsten Gateways für die Suche besser ist.

Wenn der Pfadschaltungsdienst die QoS-Parameter nicht auswerten kann, muss das wieder eine Metrik sein. Für die Konzeption der Pfadschaltung und der Gateways wird davon ausgegangen, dass das QoS-Backend eine Metrik berechnet und die QoS-Parameter unverändert weiterleitet. Die mögliche Erweiterung der Schnittstelle zu den QoS-Backends wird aber bei der Konzeption berücksichtigt und auf mögliche Einschränkungen der Erweiterbarkeit hingewiesen.

Das Pfadschaltungsprotokoll unterstützt somit den Austausch von QoS-Parametern in folgender Weise:

- Definition eines QoS-Parameters für die minimale Übertragungsrate
- Definition eines QoS-Parameters für die maximale Latenz
- Definition eines QoS-Parameters für maximalen Jitter
- Definition eines QoS-Parameters für die maximale Paketverlustrate
- Option für garantierte Ressourcen zur Einhaltung *aller* geforderten QoS-Parameter
- ID zur Identifikation vereinbarter QoS-Anforderungen
- Formalismus für die Einbettung neuer QoS-Parameter

In diesem und in Abschnitt 4.3 wurden Ressourcen und QoS-Parameter und ihr Zusammenhang mit einer Metrik bereits angesprochen. Die Definition und Berechnung der Metrik aus QoS-Parametern und Ressourcen wird im folgenden Abschnitt präzisiert.

4.4.3. Die Metrik für die Pfadsuche

Die Metrik ist ein Maß für die „Länge“ von Pfaden bzw. Pfadabschnitten in Abhängigkeit von QoS-Parametern. Das Beispiel Latenz veranschaulicht die Bedeutung: Wird 0 - 100 als Wertebereich für die Metrik definiert, so hat, bei einer maximal erlaubten Latenz von 50 Millisekunden, ein Pfadabschnitt, der eine maximale Latenz von 5 Millisekunden garantieren kann, die Länge 10. Parameter wie die Latenz, die die Metrik linear verringern, sind einfach zu handhaben. Nach 10 Pfadabschnitten mit 5 Millisekunden Verzögerung ist die maximale Pfadlänge erreicht. Im Folgenden wird diese „Länge“ präziser definiert:

Definition einer Metrik für die Pfadsuche in Abhängigkeit von QoS-Parametern

Zu jedem QoS-Parameter q muss eine Funktion f_q definiert sein, die aus einem Abschnitt l eines Pfades im eigenen Netz und dem QoS-Parameter q eine Zahl m_q berechnet, für die gilt: Wird der Abschnitt l $1/m_q$ mal durchlaufen, so wird der Parameter q (gerade noch) erfüllt. Die Metrik m eines Pfadabschnittes l bezüglich einer Menge von QoS-Parametern Q ist das Maximum aller m_q, $q \in Q$. Für eine Verbindung auf diesem Pfad mit den QoS-Parametern Q gilt dann, dass alle $q \in Q$ erfüllt werden, wenn l $1/m$ mal durchlaufen wird. Die Metrik eines Pfades, d. h. von mehreren zusammenhängenden Pfadabschnitten $l_1 ... l_n$ ist $1 - (\sum_{i=1}^{n} l_i)$. Ist die Metrik eines Pfades negativ, so werden die QoS-Parameter nicht bzw. nicht sicher erfüllt.

Informell gibt die Metrik die Länge des bisherigen Pfades als Bruchteil einer möglichen (durch die QoS-Parameter begrenzten) Gesamtlänge an. m und m_q können für die Darstellung in den Pfadschaltungsnachrichten mit einer Zahl (dem höchsten Wert aus dem Wertebereich der Metrik) multipliziert werden, in dem Beispiel oben wäre diese Zahl 100, die Länge ist dadurch in Prozent der Gesamtlänge angegeben.

Diese Metrik erlaubt die einfache Pfadsuche, die in Abschnitt 4.3.2 beschrieben wurde und unterstützt mehrere QoS-Parameter. Theoretisch werden zwar beliebig viele QoS-Parameter unterstützt, allerdings skaliert diese Metrik nicht besonders gut mit voneinander unabhängigen QoS-Parametern, wie das folgende Beispiel zeigt:

Eine Pfadsuche durchläuft AS1, AS2, AS3 und AS4. AS1 und AS3 berechnen für Latenz und Paketverlustrate jeweils die Metriken $m_L = 10$ und $m_P = 30$ und daraus das Maximum bzw. die lokale Metrik $m = 30$. AS2 und AS4 berechnen für Latenz und Paketverlustrate jeweils die Metriken $m_L = 30$ und $m_P = 10$ und daraus das Maximum bzw. die lokale Metrik $m = 30$. Der gesamte Pfad hat also eine Metrik von $100 - 120 = -20$ und erfüllt daher scheinbar nicht die QoS-Parameter, obwohl mit Wissen aller lokalen Metriken eine Pfadmetrik von $100 - 80 = 20$ berechnet werden würde. Diesen Pfad würde die Pfadsuche auf Grundlage der oben definierten Metrik nicht finden.

Sind die QoS-Parameter nicht unabhängig, so tritt ein solcher Fall selten auf. Für die vier in das Pfadschaltungsprotokoll integrierten Parameter, *minimale Übertragungsrate*, *maximale Latenz*, *maximaler Jitter* und *maximale Paketverlustrate*, ist die Metrik in vielen Fällen geeignet. Die Metrik der Übertragungsrate ist immer 100 oder 0 (entweder steht die Übertragungsrate zur Verfügung oder nicht), und die Metriken der anderen Parameter sind in der Regel nicht unabhängig (die Überlastung von Routern führt zu einer höheren Paketverlustrate und einer höheren Latenz und, wenn Pakete wegen Überlastung teilweise umgeleitet werden, zu höherem Jitter). Wo diese Annahme nicht zutrifft, oder falls weitere QoS-Parameter eingesetzt werden sollen, die diese Metrik unbrauchbar machen, so ist eine andere Lösung sinnvoll. Vorschläge dazu werden am Ende dieses Abschnittes gemacht.

Eine Funktion zur Berechnung einer Metrik bezüglich eines QoS-Parameters ist eine *gute Metrik-Funktion*, wenn mit dieser Funktion alle Pfade für eine Verbindung gefunden werden, die nur diesen QoS-Parameter verwendet.

Die Berechnung einer guten Metrik für die Latenz ist einfach. Die Verzögerung im eigenen AS wird durch die im QoS-Parameter angegebene maximale Latenz geteilt (und gegebenenfalls für die Übertragung mit dem Maximalwert der Metrik multipliziert). Der Grund für die Einfachheit der Berechnung liegt darin, dass sich die Latenzen in den einzelnen autonomen Systemen zu der Gesamtlatenz addieren.

Die Berechnung einer guten Metrik für Paketverlustrate und Jitter komplizierter. Sind Werte für Jitter im eigenen AS nach der Definition in Abschnitt 4.4.2 angegeben, so lassen sich die Jitter-Werte der durchlaufenen autonomen Systeme addieren, und man erhält, wie im Abschnitt 4.3.3 bereits beschrieben, einen Worst-Case-Jitter. Die Wahrscheinlichkeit, dass der Worst-Case eintritt, ist aber nicht bekannt und möglicherweise extrem gering. Ist der Worst-Case-Jitter kleiner, als der durch den QoS-Parameter definierte Wert, so erfüllt der Pfad die QoS-Anforderungen, d. h. man kann mit dieser Berechnung geeignete Pfade finden. Möglicherweise findet man aber mit einer besseren bzw. realistischeren Berechnung weitere Pfade. Auf der Grundlage dieser Worst-Case-Berechnung

berechnet sich die (nicht unbedingt „gute“) Jitter-Metrik genauso wie für die Latenz: Der Jitter im eigenen AS wird durch den maximal erlaubten Jitter geteilt.

Bei der Paketverlustrate verhält es sich ähnlich: Zufällige Paketverluste ereignen sich selten in allen autonomen Systemen gleichzeitig. Hier wird ein Beispiel für eine naive Metrik-Funktion, die nicht „gut“ im Sinne der oben gegebenen Definition ist, angeführt. Ein Pfad besteht aus 5 Abschnitten und in jedem Abschnitt geht 1/10 der Pakete verloren. Dann ist die effektive Paketverlustrate $1 - 0,9^5 = 0,40951$. Würde die Paketverlustrate einfach addiert, so ergäbe sich eine Paketverlustrate von $5 * 0,1 = 0,5$. Eine Metrik lässt sich auf dieser Grundlage folgendermaßen berechnen: Die Paketverlustrate im eigenen AS wird durch die maximale Paketverlustrate geteilt. Die Funktionswerte erfüllen die Definition einer Metrik, die Funktion erfüllt aber nicht die Definition einer guten Metrik-Funtion.

Suchergebnisse gehen also durch schlechte Metrik-Funktionen und durch paarweise unabhängige, von der Pfadlänge abhängige, QoS-Parameter verloren.

An dieser Stelle sei darauf hingewiesen, dass die Schwierigkeiten *nicht in der Berechnung der Metrik* liegen, sondern in der Berechnung des eigenen Anteils am „Verbrauch“ der QoS-Parameter. Diese Berechnung muss auf jeden Fall erfolgen, es stellt sich nur die Frage, wie sie kommuniziert und ausgewertet wird, und genau dafür ist die Metrik gedacht. Ohne eine Metrik müsste der Algorithmus für die Pfadsuche jeden QoS-Parameter kennen und auswerten; jeder dieser Parameter müsste über eine (entsprechend komplizierte) Schnittstelle mit den QoS-Backends für die Pfadsuche kommuniziert werden (bei dieser Kommunikation käme höchstwahrscheinlich auch eine Art Metrik zum Einsatz).

Die Metrik bedeutet also eine enorme Vereinfachung des Problems auf Kosten der Präzision, d. h. der Ergebnismenge. Eine präzise Berechnung des „Verbrauchs“ von QoS-Parametern auf dem ganzen Pfad ist aber trotzdem möglich und kann in Form von dafür definierten QoS-Parametern geschehen. Ein QoS-Parameter lautet dann nicht mehr „maximale Latenz = 50 ms“ sondern beispielsweise „Latenzverbrauch bisher = 30 ms, Verbrauch in diesem AS = 10 ms, Latenzobergrenze für den Pfad = 50 ms“.

Wird die Metrik nicht zur Pfadsuche verwendet oder ist kein QoS-Parameter von der Länge des Pfades abhängig, so übernimmt die Metrik die Rolle eines Hop-Limits und wird auf jedem Gateway um einen Mindestbetrag verringert. Man könnte sagen, dass ein Hilfs-QoS-Parameter eingeführt wird, der die Anzahl an beteiligten Gateways begrenzt. Diese Rolle der Metrik wird durch einen anderen Wertebereich erkennbar gemacht.

Ein Kompromiss zwischen der Komplexität, die eine Berücksichtigung und Anpassung aller QoS-Parameter für die Pfadsuche auf jedem Gateway mit sich bringt, und den schlechteren Ergebnissen, die die Reduzierung aller QoS-Parameter auf eine Metrik zur Folge hat, wäre es, die oben definierte Metrik für jeden Parameter zu berechnen und diese Metrik in den Pfadschaltungsnachrichten an die QoS-Parameter „anzuhängen“. Bezogen auf das Latenzbeispiel sähe ein Parameter dann so aus: „Latenz = 50 ms, Latenzmetrik für den gesamten bisherigen Pfad = 3/5, Latenzmetrik für den lokalen Pfadabschnitt = 1/5“. Der Algorithmus für die Pfadsuche müsste die QoS-Parameter nicht kennen, sondern könnte mit Metrik-Vektoren arbeiten, die transparent von den QoS-Backends berechnet würden. Die Vektoren würden allerdings für jede Verbindung, je nach geforderten QoS-Parametern, anders aussehen. Der Algorithmus für die Pfadsuche wäre davon nicht betroffen, wenn er immer Vektoren verwendete, die alle von den lokalen QoS-Backends unterstützten QoS-Parameter enthielten und nur in den für die aktuelle Pfadsuche benötigten Feldern gültige Werte eingetragen wären. Die von der Verbindung nicht benötigten QoS-Parameter bzw. Felder im Vektor müssten dann einen Wert bekommen, der die Pfadsuche nicht beeinflusst. In den Pfadschaltungsnachrichten müssten diese QoS-Parameter wieder entfernt werden.

Diesen Kompromiss könnte man verhältnismäßig einfach mit neu definierten eingebetteten QoS-Parametern nachrüsten, wenn die einfache Metrik sich als zu ungenau erwiese.

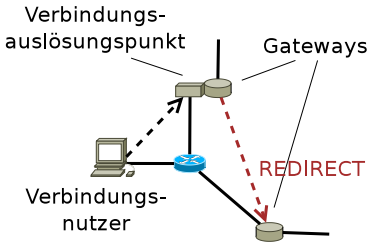


Abbildung 4.10.: Umleitung einer Verbindungsanfrage

4.5. Routing, Rewriting und Adressen

In Abschnitt 4.3.3 wird gefordert, dass der Dienstnutzer nur einen Zugangspunkt für die Verbindungsauslösung kennen muss. Falls der Zugangspunkt auf einem Gateway läuft und dieses Gateway, wie in Abbildung 4.10, feststellt, dass für eine Verbindung in diese Richtung ein anderes Gateway zuständig ist, so wird ein Umleitungsmechanismus benötigt. Das Gateway bestätigt dem Auslösungsdienst nicht die Annahme der Verbindung sondern schickt eine Umleitungsnachricht mit der Adresse des Gateways, das für die Verbindung zuständig ist. Auf diese Weise müssen Verbindungsauslöser und Verbindungsauslösungsdienst nur ein Gateway kennen und insbesondere nicht herausfinden, welches Gateway für welche Verbindungsendpunkte zuständig ist.

Bei der Pfadschaltung kommen folgende Adressen zum Einsatz, deren Verwendung und Verwaltung in diesem Abschnitt dargestellt werden:

1. Eine Adresse für die Verbindungsauslösung
2. Die Adressen der Verbindungsendpunkte
3. Die Adressen der Nachbar-Gateways
4. Der QoS-Adress-Pool (bzw. das QoS-Subnetz)
5. Die Rewriting-Adressen

4.5.1. Gateway-Lookup

Die Gateways müssen die Adresse eines Verbindungsendpunktes mit Gateways verknüpfen, die als nächstes Gateway auf dem Pfad in Frage kommen. Diese Verknüpfung kann auf unterschiedliche Weise geschehen.

Die Verbindungsendadresse kann einem AS zugeordnet werden und es kann überprüft werden, ob dieses AS ein Gateway betreibt. Dazu ist es notwendig, eine Liste mit autonomen Systemen zu pflegen, die Gateways betreiben. Betreibt das AS kein Gateway, so kann die Verbindung abgebrochen werden. Bei entfernten Gateways kann so eine vergebliche Verbindungssuche früh abgebrochen werden, das Wissen über Gateways in entfernten autonomen Systemen ist aber nicht notwendig für die Pfadschaltung. Die Kenntnis der Gateways in benachbarten Systemen ist aber unbedingt notwendig. Verfügt das autonome System, in dem der Verbindungsendpunkt liegt, also nicht über ein Gateway, so wird das spätestens in den benachbarten autonomen Systemen festgestellt und die Suche abgebrochen.

Betreibt das autonome System des Verbindungsendpunktes ein Gateway oder liegen darüber keine Informationen vor, so müssen die nächsten für einen Pfad in Frage kommenden Gateways gefunden werden. Informationen zum normalen Routing in das AS des Verbindungsendpunktes sind in jedem AS verfügbar, auf diesem Weg kann das Gateway benachbarte autonome Systeme ermitteln, die als Transit-AS in Frage kämen. Das Gateway muss noch herausfinden, ob in dem potentiellen Transit-AS ein Gateway steht. Ist das der Fall, so ist ein potentielles Gateway für den Pfad gefunden.

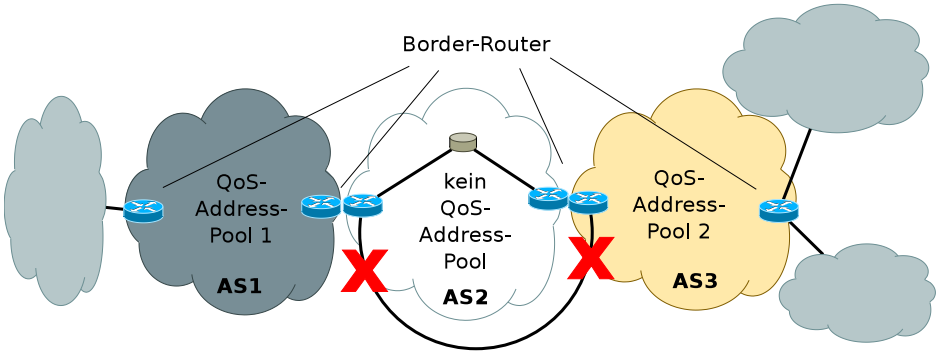


Abbildung 4.11.: AS2 liegt als reines Transit-AS zwischen AS1 und AS3 und routet Adressen aus dem QoS-Address-Pool von AS1 direkt nach AS3 und umgekehrt; das Gateway in AS2 ist nur für die Ressourcenverwaltung notwendig, Address-Rewriting wird nicht benötigt

Die Information, welche Netze bzw. Endpunkte über welches AS geroutet werden, kann von Border-Routern bzw. BGP-Speakern bezogen werden. Die Zuordnung von benachbarten autonomen Systemen zu Gateway-Adressen muss das Gateway selbst verwalten, es wäre aber möglich, dass die Nachbarn ihre Gateways auch über BGP austauschen.

In der beschriebenen Weise findet das Gateway nur nächste Gateways in autonomen Systemen, die Peers oder Provider des eigenen AS sind; das schränkt die Pfadsuche ein. Ein benachbartes AS, das weder Peer noch Provider ist, könnte ein guter Kandidat für das nächste Gateway auf dem Pfad sein, aber nicht für das normale Routing in das Ziel-AS in Frage kommen. Informationen dieser Art können auf andere Weise, z. B. von Hand, gepflegt werden. Da bei der Pfadschaltung Adressen aus dem Nachbar-AS verwendet werden, spielen die Peering-Verträge hier keine Rolle.

Zusätzlich zu den BGB-Informationen können also eigene Tabellen gepflegt werden, die entfernte autonome Systeme mit benachbarten für die Pfadschaltung verknüpfen (und diese Verknüpfung eventuell bewerten). Die automatisch vorhandenen und gepflegten Informationen von einem BGP-Speaker können mit den selbstgepflegten Tabellen in einem Verzeichnisdienst zusammengeführt werden, auf den die eigenen Gateways zugreifen. In der Testphase, solange nur wenige autonome Systeme und Gateways im Einsatz sind, reichen die selbstgepflegten Informationen aus. Eine Schnittstelle zu BGP wird nötig, sobald die manuelle Pflege der Verzeichnisse zu aufwändig wird.

4.5.2. Determinisierung des Routing

Nachdem die Suche der geeigneten Gateways zur Pfadschaltung behandelt wurde, geht es im folgenden Absatz um die Pfadschaltung. Die Einhaltung eines Pfades wird durch Determinisierung des Routing erreicht, die Determinisierung des Routing wiederum durch Zuordnung von Rewriting-Adressen zu einer Verbindung für das Address-Rewriting.

Bis auf Ausnahmen (Transit-AS ohne eigene Verbindungsendpunkte, siehe Abbildung 4.11) muss jedes Gateway über einen Pool von Adressen verfügen, die für das Rewriting verwendet werden können. Eine Adresse aus diesem Pool dient dazu, das Routing von der (oder einer) AS-Grenze bis zu einem Rewriting-Router zu determinisieren. Auf dem Pfad zwischen der bzw. den Grenzen (Border-Routern) bis zum Rewriting-Router müssen die für die Pfadschaltung vorgesehenen QoS-Backends verfügbar sein. Je nach Größe und Topologie eines autonomen Systems kann ein QoS-Subnetz und ein Rewriting-Router in der topologischen Mitte ausreichen, um diese Anforderungen zu erfüllen. Bei einem größeren autonomen System können für jeden Rewriting-Router eigene QoS-Netze verwendet werden.

Die *QoS-Netze* müssen *nicht* mit den *QoS-Address-Pools* übereinstimmen. Die QoS-Netze werden Rewriting-Routern zugeordnet, die Address-Pools den Gateways. Jedes Gateway kann einen Teil eines

QoS-Netzes zur Verwaltung zugeteilt bekommen, sodass der Address-Pool eines Gateway Adressen aus allen QoS-Netzen des autonomen Systems enthält. Es ist aber auch möglich, dass QoS-Address-Pool und QoS-Netz übereinstimmen, insbesondere dann, wenn ein Gateway auch die Rolle des Rewriting-Router einnimmt. Die Verteilung der QoS-Netze auf die Gateways hängt von den Pfaden diese Gateways schalten sollen.

Da die Rewriting-Router die Zielpunkte für die QoS-Netze sind, müssen sie an geeigneten Punkten (zwischen zwei Nachbar-Gatways) auf den QoS-fähigen Netzabschnitten platziert werden. Das Routing vom Rewriting-Router zu den benachbarten autonomen Systemen muss ebenfalls auf den QoS-fähigen Netzabschnitten erfolgen. Da Informationen über die QoS-Netze der Nachbarn nicht so leicht zu pflegen sind, wie die eigenen, ist es von Vorteil, die Rewriting-Router möglichst nah am Border-Router zu platzieren. Ideal wäre es, wenn die Border-Router auch Rewriting-Router wären, da dann das Routing im eigenen AS nur von den eigenen QoS-Netzen abhängt und die neuen Adressen erst im anderen AS für das Routing verwendet werden. Allerdings bräuchte man dann ein QoS-Netz für jeden Border-Router.

Falls die Border-Router nicht die Rewriting-Router sind, hängt ein Teil des Pfades innerhalb des eigenen AS von der Rewriting-Adresse ab, die das nächste Gateway bei der Pfadschaltung zur Verfügung stellt. Für diesen Teil des Pfades müssen Ressourcen reserviert oder vorgemerkt werden und dazu muss der Teil des Pfades bekannt sein. Da zum Zeitpunkt der Pfadsuche noch keine Rewriting-Adressen aus den benachbarten Gateways zur Verfügung stehen, muss der entsprechende Teil des Pfades anders ermittelt werden. Ist bekannt, aus welchem QoS-Netz die Rewriting-Adresse bei der Pfadschaltung stammen wird, so kann der Teil des Pfades auf diese Weise ermittelt werden. Ist nicht bekannt, aus welchem QoS-Netz die Rewriting-Adresse stammen wird, so kann die Gateway-Adresse zur Ermittlung des Pfad-Abschnittes verwendet werden. Das wird am einfachsten dadurch erreicht, dass das Gateway über eine Adresse aus dem eigenen QoS-Netz angesprochen wird. Diese Vorgehensweise funktioniert bei einfachen Konstellationen, wo das Gateway nur ein QoS-Netz verwaltet oder am Übergang zwischen den autonomen Systemen nur ein Border-Router eingesetzt wird. Ist Letzteres der Fall, so steht der Teil des Pfades vom Rewriting-Router bis zum Border-Router ohnehin fest.

Auf den Rewrite-Routern wird die Empfängeradresse der IP-Pakete einer Verbindung ausgetauscht. Bei Bedarf wird auch die Absenderadresse ausgetauscht. Das ist nur möglich, wenn bei der Pfadschaltung ein bidirektionaler Pfad geschaltet wurde. Die Absenderadresse der Verbindung wird im letzten Rewriting-Router vor der Vermittlung an den Endpunkt ausgetauscht. Es ist nur nötig, das vorher zu tun, falls die Absenderadresse für etwas anderes, beispielsweise für Demultiplexing, benötigt wird. Sollte ein Gateway keine QoS-Adressen mehr zur Verfügung haben, kann es mehreren Verbindungen dieselbe QoS-Adresse zuordnen und diese Verbindungen über einen Pfad „multiplexen“.

4.5.3. Transparentes Multiplexing von Verbindungen

Der Einsatz von Verbindungs-Multiplexing ist transparent möglich, wenn folgende Bedingungen erfüllt sind:

1. Auf dem Rewriting-Router, auf dem das Multiplexing stattfindet, wird überwacht, dass keine Verbindung die Ressourcen einer anderen Verbindung einschränkt, d. h. sobald eine Verbindung mehr Ressourcen verbraucht, als ihr zustehen, wird der Ressourcenverbrauch dieser Verbindung gezielt gedrosselt; da das Multiplexing über mehrere autonome Systeme hinweg durchgeführt werden kann, reicht es nicht, auf Engpässe im eigenen AS zu reagieren.
2. Auf dem Rewriting-Router in dem AS, das die Rewriting-Adressen für das Multiplexing vergibt, ist Demultiplexing möglich und die Informationen für das Demultiplexing (Mapping von Absenderadressen auf Rewriting-Adressen) werden von dem Gateway in diesem AS zur Verfügung gestellt.

Das Gateway, das die Rewriting-Adressen für das Multiplexing vergibt, muss sich vergewissern, dass der Rewriting-Router im vorherigen AS die Fähigkeiten zur Drosselung der Ressourcen hat und auf dem eigenen Rewriting-Router das Demultiplexing schalten. Erkennt ein Gateway das Multiplexing

des nächsten Gateways, so kann es das Multiplexing an das vorherige Gateway weiterleiten, wieder vorausgesetzt, dass der dortige Rewriting-Router gegebenenfalls die Ressourcendrosselung durchführt.

Ein weiteres Feature für Verbindungen, das von den Fähigkeiten der Rewriting-Router abhängt, ist Multicast. Kann der Rewriting-Router IP-Pakete duplizieren und das Duplikat an eine andere Adresse versenden, so kann diese Fähigkeit für Multicast-Verbindungen genutzt werden. Da die Multicast-Fähigkeiten der in der Implementierung eingesetzten Backends aber noch nicht ausgereift sind, wird Multicast in Kapitel 7 (Ausblick) behandelt.

Die Konzepte dieses Abschnitts sind im Folgenden in Stichpunkten zusammengefasst:

- Redirect-Nachricht, die eine Verbindungsanfrage an ein anderes Gateway umleitet
- Zuordnung von Verbindungsendpunkten zu Gateways auf jedem Gateway (Verzeichnisdienst)
- Zuordnung von Verbindungsendpunkten zu autonomen Systemen (global)
- Kenntnis der benachbarten Gateways
- Zuordnung von den benachbarten autonomen Systemen zu dort aufgestellten Gateways
- Zuordnung von entfernten zu benachbarten autonomen Systemen
- QoS-Address-Pool für (fast) jedes Gateway
- Rewriting-Router auf QoS-fähigem Netzabschnitt
- Routing der QoS-Adressen über den Rewriting-Router
- Border-Router als Rewriting-Router oder
- Routing der benachbarten Gateways über dieselben Border-Router, wie die dazugehörigen QoS-Netze oder
- Kenntnis aller QoS-Netze des Nachbarn und Einsatz bei der Pfadsuche
- Routing der Nachbar-QoS-Netze (soweit bekannt und wenn die Border-Router nicht Rewriting-Router sind) über QoS-fähige Netzabschnitte
- Demultiplexing als Feature für die Rewriting-Router
- Ressourcendrosselung einzelner Verbindungen als Feature der Rewriting-Router
- Multiplexing von Verbindungen
- Multicast abhängig von den Features der Rewriting-Router

Im folgenden Abschnitt wird analysiert, welche Fehler auftreten können, wie auf sie reagiert werden kann oder wie sie vermieden werden können.

4.6. Fehlerbehandlung

Die am Anfang des Kapitels vorgestellten funktionalen Komponenten sind alle kritisch für die Pfadschaltung, aber auch der normale Routing-Betrieb kann durch Fehler dieser Komponenten beeinträchtigt werden. Auf Abbildung 4.12 sind einige Fehler und Stellen, an denen sie auftreten können, gezeigt. In Abschnitt 4.6.1 werden zuerst die Fehler betrachtet, die bei Auf- und Abbau sowie Nutzung einer Verbindung auftreten können, Abschnitt 4.6.2 analysiert dann Fehler, die den übergeordneten Betrieb, nicht nur der Pfadschaltung, betreffen. Dabei werden der Reihe nach die Komponenten und Schnittstellen aus Abbildung 4.12 auf mögliche Fehlerquellen untersucht und auf Bezeichnungen und Nummerierung im Bild Bezug genommen.

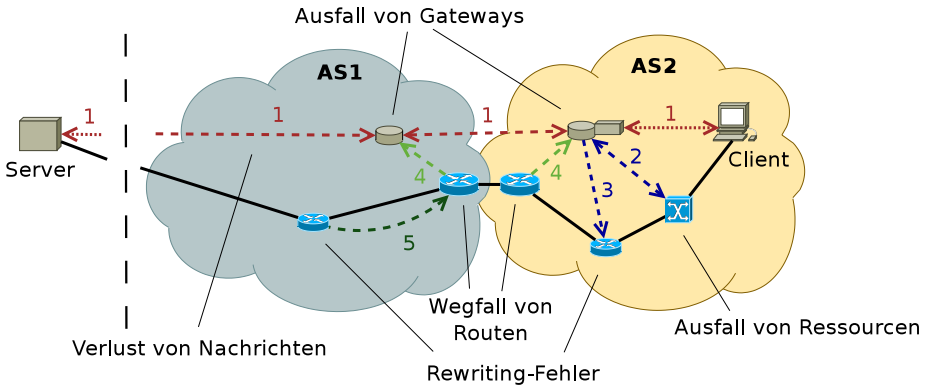


Abbildung 4.12.: Fehler können auf und zwischen funktionalen Komponenten auftreten

4.6.1. Fehler, die eine Verbindung betreffen

Pfadschaltungsnachrichten, die zwischen *Gateways* ausgetauscht werden (1) können verloren gehen. Das Versenden der Nachrichten über zuverlässige Verbindungen (z. B. TCP) löst das Problem zwar übertragungsseitig. Ob die Nachrichten aber von einem Pfadschaltungsdienst verarbeitet wurden, ist damit noch nicht gesichert. Daher ist eine Bestätigungsnachricht des Pfadschaltungsdienstes notwendig. Die Bestätigungsnachricht bestätigt auch die Übertragung, daher kann auf TCP verzichtet werden. Methoden zur Bestimmung der Wartezeit für die Bestätigung und gegebenenfalls die erneute Übertragung müssen in den Pfadschaltungsdienst integriert werden.

Während der Pfadsuche und Schaltung kann es zu Fehlern in der Berechnung der Ressourcen bzw. zu Engpässen kommen (2). Fehler werden durch Überbuchung der Ressourcen beim Vormerken bewusst in Kauf genommen, können aber auch andere Ursachen haben. Diese Fehler können im Rahmen der Pfadschaltung nur im eigenen Netz behandelt werden, ob die anderen beteiligten autonomen Systeme die Ressourcen korrekt zuordnen, muss auf andere Weise festgestellt werden (Auditing- und Monitoring von Testverbindungen etc.). Wird ein Engpass während der Pfadschaltung erkannt, so muss darauf reagiert werden können. Wenn alternative Pfadabschnitte bekannt sind, mit einer Redirect-Nachricht (siehe Abschnitt 4.5), andernfalls mit einer Fehlernachricht an beide Nachbar-Gateways.

Fallen Ressourcen einer bestehenden Verbindung aus, muss das festgestellt werden können. Das bedeutet, dass die Information über reservierte Ressourcen nicht nur auf Komponenten gespeichert werden, die diese Ressourcen bereitstellen (z. B. Routern), da sonst mit deren Ausfall auch die Information verloren geht, dass der Verbindung Ressourcen fehlen! Die Aufgabe der Ressourcenüberwachung und der Meldung eines Ausfalls (bezüglich spezifischer Verbindungen) liegt bei den Resource- und QoS-Backends. Falls die Backends die Ressourcen nicht selbstständig wieder zuordnen können, müssen die Pfadschaltungsgateways die Ausfallsmeldung auswerten und entweder Ressourcen im eigenen AS neu zuordnen oder mit einer Redirect- oder Fehlernachricht an die beiden benachbarten Gateways reagieren.

Fällt ein Gateway, das für eine bestehende Verbindung zuständig ist, aus, so sind die Konsequenzen aus Sicht der Nachbarn unklar. Möglicherweise ist die Verbindung von dem Ausfall gar nicht, oder erst beim Verbindungsabbau betroffen. Da die Ressourcen in einem AS nur durch das dort stehende Gateway vorgenommen werden kann, sind die Nachbar-Gateways auf Informationen von diesem Gateway angewiesen (Keepalive-Nachrichten). Ob die Verbindung weiterhin steht, wenn das Gateway ausgefallen ist, können die Nachbar-Gateways nicht überprüfen, daher muss auf einen Gateway-Ausfall immer gleich reagiert werden. Auf der Seite des Initiator-Gateways kann ein neuer Pfad gesucht werden, auf der anderen Seite kann die Verbindung entweder gleich abgebaut werden, oder noch einige Zeit gewartet werden, ob ein Pfad über ein Ersatz-Gateway gefunden wird. Wird ein neuer Pfad z. B. über ein Ersatz-Gateway im AS des ausgefallenen Gateways gefunden, so bemerken die Enden möglicherweise gar nichts von dem Ausfall und der Umleitung.

Die Zuordnung von Rewriting-Adressen und die Aktivierung des Rewriting passieren durch den Pfadschaltungsdienst auf den Gateways (3). Fällt ein *Rewriting-Router* aus, so muss das Gateway davon erfahren und das Rewriting für die bestehenden Verbindungen wiederherstellen. Das Vorhandensein der Informationen über aktive Address-Rewritings ist dadurch gesichert, dass das Rewriting beim Verbindungsabbau wieder deaktiviert werden muss, allerdings spielt die Form der Information eine Rolle: Die Gatweays müssen die Rewriting-Adressen lokal speichern, es reicht nicht, diese Adressen auf den Rewriting-Routern zu speichern und über die Verbindungs-ID anzusprechen.

Die Erkennung unzulässiger Rewriting-Operationen auf den Rewriting-Routern würde der Vermeidung von Fehlern dienen, allerdings erhöht das die Komplexität, je nach implementierten Features, deutlich: Die Rewriting-Router müssten die QoS-Netze der Nachbarn, die zulässigen Endpunkte im eigenen AS, die Zuordnung von Adressen zu Verbindungen (für die Erkennung versehentlich doppelt verwendeter Adressen) und Verbindungsmultiplexing kennen (für das Erkennen von absichtlich doppelt verwendeter Adressen).

Ein Update der Informationen auf den Border-Routern (4) kann die Pfadschaltung ebenfalls betreffen. Durch das Address-Rewriting sind die Gateways relativ unabhängig von Verträgen (Peering und Providing), da die autonomen Systeme auf IP-Ebene scheinbar nie die Rolle eines Transit-AS einnehmen. Fallen Router aufgrund von Topologieänderungen (das schließt Router-Ausfälle ein) aus, so ist die Pfadschaltung ebenfalls betroffen. Die Gateways müssen darüber informiert werden und bestehende Verbindungen müssen entsprechend umgeleitet oder abgebrochen werden. Topologieänderungen werden aber in der Regel lokal erkannt, für entfernte Änderungen sind die dort zuständigen Gateways zuständig.

Wenn die Rewriting-Router ihre QoS-Netze nicht richtig bekanntgeben (5), so kann es sein, dass die Verbindungen ins Leere laufen, da sie den richtigen Rewriting-Router nicht erreichen und das Rewriting nicht stattfindet. Es ist nicht eindeutig, wo dieser Fehler bemerkt werden kann. Die Enden bemerken den Fehler auf jeden Fall, an anderen Stellen sieht es nur so aus, als würde die Verbindung gerade nicht verwendet. Der Router, auf dem die Pakete fälschlicherweise landen, würde den Fehler zwar auch bemerken, die Zuordnung und Behebung des Fehlers kann aber nicht durch die Pfadschaltung passieren. Für die Konsistenz der Routing-Tabellen ist die Pfadschaltung nicht zuständig.

Beim Verbindungsabbau ist für den Nutzer hauptsächlich von Interesse, dass die Verbindungsdauer korrekt abgerechnet wird, d. h. sobald der Verbindungsabbau vom Initiator-Gateway oder vom Verbindungsauslösungsdienst bestätigt wurde, ist für den Verbindungsnutzer die Verbindung beendet und Fehler können für ihn nicht mehr auftreten. Weitere Fehler beim Verbindungsabbau (Fehler jenseits des Initiator-Gateway) betreffen weniger eine einzelne Verbindung, da diese als beendet gilt, sondern den allgemeinen Betrieb und werden im nächsten Abschnitt behandelt.

4.6.2. Fehler, die den allgemeinen Betrieb beeinträchtigen

Fehler beim Verbindungsabbau können die Interaktionen (1), (2) und (3) betreffen. Wird der Verbindungsabbau von einem Gateway nicht korrekt weitergeleitet, so bleibt die Verbindung auf Abschnitten des Pfades bestehen (verwaiste Verbindung) und bindet Ressourcen. Erhält ein Gateway einen Teil der Verbindung absichtlich aufrecht (z. B. um diese Verbindung auf Kosten anderer für eigene Zwecke zu nutzen), so ist das kein Fehler und muss in einem Sicherheitskonzept behandelt werden. Scheitert die Weiterleitung am Ausfall eines Gateways, so wird die Verbindung nach einem Timeout (Ausbleiben von Keepalive-Nachrichten) beendet, daher ist hier kein neuer Mechanismus für die Erkennung ausbleibender Verbindungsabbau-Nachrichten notwendig.

Fehler in der lokalen Ausführung des Verbindungsabbaus können auch zu gebundenen Ressourcen führen, z. B. wenn die Freigabe der Ressourcen (2) fehlerhaft durchgeführt wird, wenn das Rewriting (3) nicht deaktiviert wird, oder wenn die verwendete Rewriting-Adresse nicht wieder zur Verwendung freigegeben wird. Die Behandlung dieser Fehler ist zwar sehr wichtig, muss aber, soweit sie von den QoS-Backends abhängt, individuell von den Netzbetreibern gelöst werden.

Im Folgenden werden verwaiste Ressourcen behandelt, die den Pfadschaltungsdienst betreffen: Die Information, welche Ressourcen wem gehören und ob sie noch benötigt werden, liegt (zumindest

abstrakt) bei den Gateways. Fallen Gateways aus, so sind die Informationen über verwendete Ressourcen, aktive Rewritings und verwendete Rewriting-Adressen zu einem gewissen Grad verloren und können auch nicht aktualisiert werden. Zu einem gewissen Grade bedeutet, dass die QoS-Backends und Rewriting-Router zwar wissen können, dass Ressourcen reserviert und Address-Rewritings aktiviert wurden, nicht aber deren korrekte Freigabe ausführen können. Je komplexer die Informationen sind, die Rewriting-Router und QoS-Backends speichern, umso besser lassen sich Ausfälle verwalten. Es könnten gar keine Informationen vorliegen, Ressourcen und Adressen einer Verbindung zugeordnet werden, Ressourcen und Adressen einem Gateway zugeordnet werden und Ressourcen und Adressen einer Verbindung und diese wiederum einem Gateway zugeordnet werden.

Damit bei einem Ausfall der Gateways nicht Ressourcen auf Verdacht (für wie lange?) gebunden bleiben oder alternativ alle bestehenden Verbindungen unterbrochen werden müssen, sollten alle Informationen, die zur Zuordnung und Freigabe von Ressourcen und Adressen benötigt werden, robust, d. h. in einer Form, die den Absturz des Dienstes oder des Gateways überlebt, und möglicherweise sogar redundant gespeichert werden, sodass ein Ersatz-Gateway darauf zugreifen kann.

Eine Fehlerquelle, die den Betrieb der Gateways belasten kann, sind nicht protokollkonforme oder unberechtigte Pfadschaltungsnachrichten. Auf beides sollte mit einer Fehlernachricht reagiert werden, ob bei „Wiederholungstätern“ weitere Schritte unternommen werden sollen, ist eine Streitfrage, die bei allen Internetprotokollen auftritt und unterschiedlich gelöst wird. Einerseits sollen die fehlerhaften Nachrichten die Verarbeitung der korrekten Nachrichten nicht stören, andererseits treffen Schritte, die dagegen unternommen werden, oft auch „Unschuldige“ oder können sogar gezielt missbraucht werden. Das fällt in den Bereich der Sicherheit (DoS), der im folgenden Abschnitt angesprochen wird.

4.7. Sicherheit, Logging und Accounting

Für die dienstgütespezifische Pfadschaltung ist ein Sicherheitskonzept unerlässlich, da Ressourcen reserviert und IP-Adressen ersetzt werden. Es gilt die unberechtigte Belegung von Ressourcen und das ungewollte Adress-Rewriting von beliebigen Adressen zu verhindern. Letzteres ist besonders brisant, da Address-Rewriting das Umleiten der IP-Adressen, z. B. von Banken oder Online-Shops auf eine Seite für Passwortdiebstahl ermöglicht, die Absenderadresse bei Angriffen verschleiern, oder Adressen „ins Leere“ leiten kann. Der Zugriff auf die Rewriting-Funktionen der Router muss daher sorgfältig geschützt werden, beispielsweise indem der Zugang auf einen IPSec oder SSH-Tunnel eingeschränkt wird. Zusätzlich sollten die Rewriting-Adressen überprüft werden, beispielsweise anhand einer White-List mit allen erlaubten Absender- und Empfängeradressen bzw. Netzen. Dafür müssen der eigene QoS-Address-Pool (für die Absenderadressen) und die Address-Pools der Nachbarn bekannt sein. Auch der Zugriff auf die QoS-Backends und die Ressourcenverwaltung müssen vor unberechtigtem Zugriff geschützt werden.

Um Pfadschaltungsnachrichten vertrauen zu können, muss ein Gateway

1. den Absender authentifizieren und
2. die Pfadschaltung durch diesen Absender autorisieren.

Dafür könnte ein Schlüsselpaar zur Signatur der Pfadschaltungsnachrichten verwendet werden. Die Signatur kann auch für die Authentifizierung auf den QoS-Backends und den Rewriting-Routern eingesetzt werden. Die öffentlichen Schlüssel aller Gateways bzw. zumindest der jeweiligen Nachbar-Gateways müssen bekannt sein. Werden nur die Nachbar-Gateways authentifiziert, so muss eine Chain-of-Trust zum Initiator-Gateway existieren. Sind die öffentlichen Schlüssel aller Gateways bekannt, so kann das Initiator-Gateway die globalen Teile der Pfadschaltungsnachrichten signieren. Das hätte den Vorteil, dass man den Verbindungsbesitzer zweifelsfrei feststellen kann und so ein Mittel hat, die Kosten einer Verbindung dem Verursacher zuzuordnen. Mit der Chain-of-Trust könnte man ein Abrechnungsmodell realisieren, das wie die Abrechnung für das Inter-Domain-Routing funktioniert. Man bezahlt einem benachbarten Netzbetreiber (Provider) die Übernahme von Verbindungen oder trifft Peering-Abkommen, wenn benachbarte autonome Systeme ungefähr gleichviel Last beim

jeweils anderen erzeugen. Die Gültigkeit der Nachrichten könnte durch eine Gültigkeitsdauer und einen Zeitstempel (beides signiert) begrenzt werden.

Für die Abrechnung und das stichprobenartige Überprüfen von Betrugsversuchen können Pfadschaltungsnachrichten gespeichert oder an das Initiator-Gateway gesendet werden. Die Nachrichten zur Bestätigung und Beendigung einer Verbindung können, mit signiertem Zeitstempel, als Nachweis dienen, dass die Verbindung bestanden hat bzw. dass zumindest die Nachbar-Gateways die Existenz der Verbindung bezeugt haben. Um Betrugsversuche, z. B. Replay von Pfadschaltungsnachrichten oder Manipulation der Verbindungsparameter durch ein Nachbar-Gateway, nachzuweisen, können diese Pfadschaltungsnachrichten an das Initiator-Gateway gesendet werden. Dieses kann, durch Vergleich mit den eigenen Verbindungsdaten, überprüfen, ob die Verbindungsparameter manipuliert wurden, oder, im Falle von Replay-Nachrichten, die Verbindung, zu der die Nachricht gehört, bereits den Zustand geändert hat oder schon wieder beendet wurde.

Der letzte Absatz dieses Abschnitts behandelt das IP-Spoofing, DoS und „Verbindungsdiebstahl“. Zum einen muss bei der Pfadschaltung darauf geachtet werden, dass der Verbindungsnutzer die IP-Adressen der Verbindung bzw. die Absenderadresse einer bidirektionalen Verbindung verwenden darf und die IP-Pakete nicht an einem Spoofing-Filter verworfen werden. Falls ein solcher Spoofing-Filter eingesetzt wird, so muss dieser durch die Pfadschaltung aktualisiert werden. Zum anderen muss bedacht werden, ob die Verbindung gekapert werden kann, d. h. ob ein benachbarter Endpunkt die Verbindungsadresse „snifft“ oder errät, Daten an diese Adresse sendet und damit die Ressourcen der Verbindung belegt. Das kann auf einem beliebigen Abschnitt der Verbindung passieren, daher sollte es möglich sein, das Senden an die Verbindungsadresse auf die legitimen Verbindungsnutzer einzuschränken und das Senden an die Rewriting-Adressen auf die Rewriting-Router zu beschränken. IP-Spoofing betrifft auch DoS-Angriffe auf Gateways: Wird ein Gateway mit fehlerhfaten Nachrichten bombardiert, so könnte es den Absender auf IP-Ebene ignorieren (die Überprüfung von Signaturen und die Auswertung der Nachrichten belasten das Gateway). Haben die fehlerhaften Nachrichten aber einen falschen Absender, kann man auf diesem Weg die Kommunikation zwischen Gateways stilllegen. Das Filtern des IP-Verkehrs zwischen den Gateways muss also an anderer Stelle erfolgen und kann nicht auf den Gateways gelöst werden.

Im Folgenden sind die Sicherheitsaspekte, die die Pfadschaltung umsetzen sollte, in Stichworten zusammengefasst:

- Authentifizierung der Absender von Pfadschaltungsnachrichten
- Autorisierung der Absender zur Pfadschaltung
- Authentifizierung des Initiator-Gateway oder Chain-of-Trust zum Initiator-Gateway
- Signierter Zeitstempel in Pfadschaltungsnachrichten
- Signierte Gültigkeitsdauer in Pfadschaltungsnachrichten (zusammen mit Zeitstempel)
- Logging von Bestätigungs- und Beendigungsnachrichten (mit signiertem Zeitstempel) für die Abrechnung
- Optionales Prüfen von Pfadschaltungsnachrichten durch das Initiator-Gateway
- Anpassung von Spoofing-Filtern an die Verwendung von Verbindungsadressen
- Senden an Verbindungs- und Rewriting-Adressen nur von berechtigten Punkten im Netz
- Senden von Pfadschaltungsnachrichten nur von berechtigten Punkten im Netz (DoS-Vorbeugung)

Dieser Abschnitt beschränkt sich auf Hinweise und Skizzen, die Ausarbeitung von Sicherheitskonzepten für die Pfadschaltung liegt außerhalb dieser Arbeit. Die Konzeption der Gateways ist somit abgeschlossen und die Grundlage für den Entwurf eines Protokolls für die Pfadschaltung im nächsten Kapitel geschaffen.

5. Das Path-Parameter-Control-Protocol

In diesem Kapitel wird ein Protokoll für die dienstgütespezifische, inter-organisationale Pfadschaltung im Internet entworfen, das Path-Parameter-Control-Protocol. Um einen ungetrübten Blick auf die Kernfunktionalität des Protokolls zu haben, wird in Abschnitt 5.1 vorerst ein Protokoll konzipiert, das von idealisierten Bedingungen ausgeht und Probleme, Fehler und Sonderfälle, die in der Realität auftreten, ausblendet. Dieses Konzeptprotokoll wird in den darauf folgenden Abschnitten für den Einsatz unter realen Bedingungen erweitert, unter Berücksichtigung der Konzepte aus Kapitel 4.

5.1. Entwurf eines Konzeptprotokolls

Die Aufgabe des Protokolls ist es, einen Pfad von Ende zu Ende über eine Reihe von Gateways zu schalten, der spezifischen QoS-Anforderungen genügt. Die dafür notwendigen Protokollnachrichten und ihr Inhalt werden in diesem Abschnitt beschrieben, dabei wird von folgenden idealisierenden Bedingungen ausgegangen:

- Nachrichten werden immer korrekt übertragen und ausgewertet
- Es wird nicht mehr als eine Verbindung gleichzeitig bearbeitet
- Das beste nächste Gateway (der nächste Knoten auf dem Pfad) in Richtung Endpunkt ist bekannt
- Gateways sind die einzigen Hops zwischen den Endpunkten
- Ressourcen und Adressen sind immer verfügbar
- Die Ressourcen werden sofort reserviert
- Jeder Verbindungsaufbauwunsch ist autorisiert
- Keine Ausfälle von Ressourcen oder Diensten
- Die Endpunkte und die Gateways kommunizieren über dasselbe Protokoll (das Konzeptprotokoll)

Aus diesen Bedingungen folgt, dass keine Suche nötig ist. Das nächste Gateway (in Abhängigkeit von der Senke) ist bekannt und verfügt über ausreichend Ressourcen. Da Fehler in der idealen Welt nicht auftreten, sind auch keine Fehler- und Bestätigungsnachrichten nötig.

Bevor die Nachrichten des Konzeptprotokolls definiert werden, muss noch geklärt werden, welche Informationen für den Verbindungsauf- und -abbau benötigt werden und wann sie übertragen werden. Jede Pfadschaltungsnachricht hat einen Absender (das ist der Verbindungsauslöser oder ein Gateway) und einen Empfänger (das ist ein Gateway oder der Verbindungsauslöser). Während des Verbindungsaufbaus wird für die Ermittlung des nächsten Gateways bzw. zur Überprüfung, ob ein Verbindungsendpunkt im eigenen AS liegt, die Adresse des *Verbindungsendpunktes* benötigt. Ist die Verbindung unidirektional, so ist das die Senke, die Adresse der Quelle wird nicht benötigt.

Eine *bidirektionale* Verbindung muss während des Verbindungsaufbaus als solche gekennzeichnet werden. Bei bidirektionalen Verbindungen ist Address-Rewriting in beide Richtungen erforderlich. Für das Address-Rewriting in Richtung „Quelle“ muss der Verbindungsauslöser dem ersten Gateway (dem Initiator-Gateway aus Kapitel 4) die Adresse des ersten Verbindungsendpunktes, d.h. des Endpunktes, der bei einer unidirektionalen Verbindung die Quelle gewesen wäre, mitteilen. Die Gateways müssen für das Rewriting in Richtung „Quelle“ eine Rewriting-Adresse an das nächste (!) Gateway

CONNECT
Absenderadresse: Gateway oder Verbindungsauslöser
Empfängeradresse: Gateway
Verbindungsendpunkt 1 bzw. Quelle (optional)
Verbindungsendpunkt 2 bzw. Senke
Verbindungsoptionen (optional): unidirektional, bidirektional, garantierte Ressourcen, Multiplexing
Rewriting-Address in Richtung Quelle (optional)
QoS-Parameter

ESTABLISHED
Absenderadresse: Gateway
Empfängeradresse: Gateway oder Verbindungsauslöser
Rewriting-Address in Richtung Senke

DISCONNECT
Absenderadresse: Gateway oder Verbindungsauslöser
Empfängeradresse: Gateway

Abbildung 5.1.: Pfadschaltungsnachrichten des Konzeptprotokolls für den Verbindungsaufbau

senden. Eine Rewriting-Adresse für das Rewriting in Richtung Senke muss dem vorherigen Gateway mitgeteilt werden.

Für die Reservierung von Ressourcen werden die geforderten *QoS-Parameter* benötigt. Weitere Verbindungsparameter, die während des Verbindungsaufbaus übertragen werden müssen, sind die *Anforderung von garantierten Ressourcen* (siehe Kapitel 4 Abschnitt 4.4.2) und Multiplexing (siehe Kapitel 4 Abschnitt 4.5.3).

Diese Informationen werden im folgenden Abschnitt Nachrichten des Konzeptprotokolls zugeordnet.

5.1.1. Grundlegende Abläufe

Das Konzeptprotokoll kommt mit drei Nachrichten aus (Abbildung 5.1). Die Nachricht `CONNECT` für den Verbindungsaufbau, die Nachricht `ESTABLISHED` für die Bestätigung des Verbindungsaufbaus und die Nachricht `DISCONNECT` für den Verbindungsabbau.

Abbildung 5.2 zeigt einen Verbindungsaufbau über das Konzeptprotokoll; die internen Aktionen, die auf dem Gateway in der Mitte dargestellt sind, finden auch auf allen anderen Gateways statt:

1. Die Quelle sendet eine `CONNECT`-Nachricht an das Initiator-Gateway mit den Verbindungsendpunkten „Quelle“ und „Senke“, der Option „bidirektional“, der eigenen Adresse als Rewriting-Adresse und den QoS-Parametern.
2. Das Initiator-Gateway sendet die `CONNECT`-Nachricht an das nächste Gateway, ersetzt vorher die Rewriting-Adresse durch eine eigene und passt die QoS-Parameter an.
3. Auf jedem Gateway werden nach dem Eintreffen der `CONNECT`-Nachricht Ressourcen entsprechend den QoS-Parametern reserviert und berechnet, wie die QoS-Parameter für die weitere Suche angepasst werden müssen (siehe Kapitel 4).
4. Wenn die Verbindung bidirektional ist, wird auf jedem Gateway außer dem letzten eine Rewriting-Adresse für die `CONNECT`-Nachricht ermittelt und das Address-Rewriting aktiviert: Pakete an die Rewriting-Adresse, die mit der `CONNECT`-Nachricht vom vorherigen Gateway eingetroffen ist, werden an die Rewriting-Adresse adressiert, die auf diesem Gateway ermittelt wurde und mit der `CONNECT`-Nachricht an das nächste Gateway gesendet wird.
5. Das Gateway leitet die `CONNECT`-Nachricht weiter, nachdem es die Rewriting-Adresse durch eine aus dem eigenen QoS-Address-Pool ersetzt und die QoS-Parameter entsprechend den Berechnungen des Resource-Backends angepasst hat.

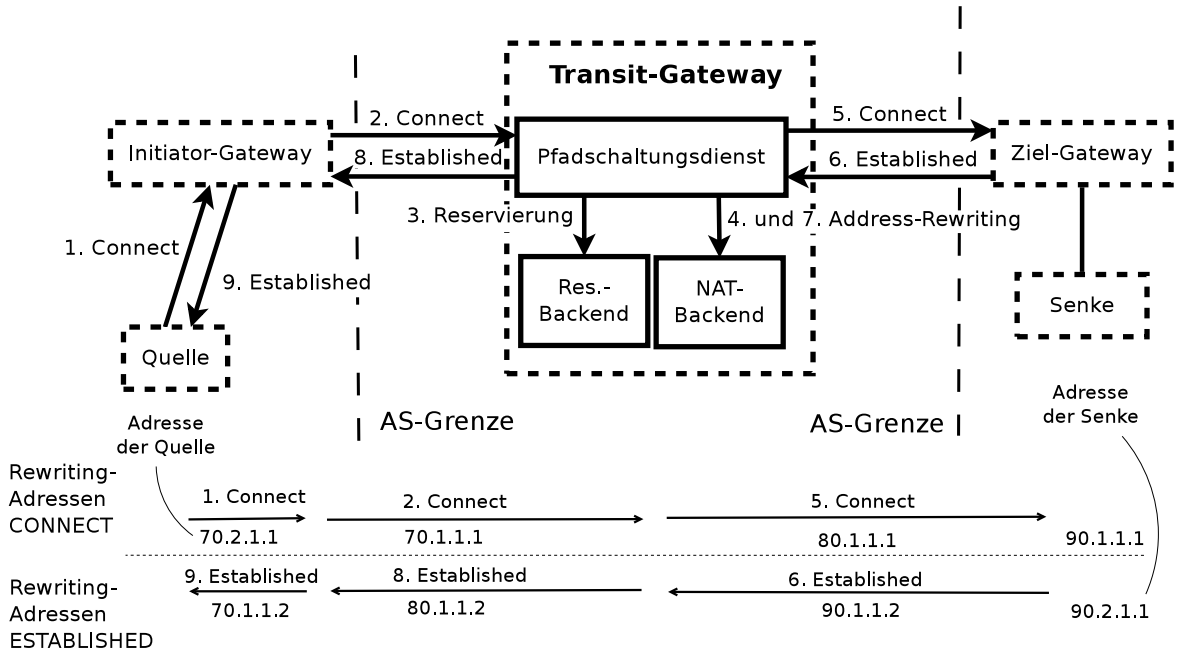


Abbildung 5.2.: Ein Verbindungsaufbau, gesteuert durch das Konzeptprotokoll; das Gateway in der Mitte zeigt stellvertretend Vorgänge in Transit-Gateways, die Vorgänge in Initiator- und Ziel-Gatway verlaufen analog

6. Das Gateway stellt fest, dass es das letzte auf dem Pfad ist, und sendet die `ESTABLISHED`-Nachricht an das vorherige Gateway; das Rewriting wird wie in den Schritten 4 und 7 aktiviert, allerdings wird die Absenderadresse der Pakete an die Rewriting-Adresse, die in Schritt 4 ermittelt wurde, durch die Rewriting-Adresse, die in Schritt 7 ermittelt wurde, ersetzt und statt der Rewriting-Adresse, die mit der `ESTABLISHED`-Nachricht eintreffen würde (die Senke sendet keine `ESTABLISHED`-Nachricht), wird die Adresse der Senke verwendet.

7. Auf jedem Gateway wird eine Rewriting-Adresse aus dem eigenen QoS-Address-Pool für die `ESTABLISHED`-Nachricht ermittelt und das Address-Rewriting aktiviert: Pakete an die Rewriting-Adresse, die mit der `ESTABLISHED`-Nachricht an das vorherige Gateway (links) versendet werden, werden an die Rewriting-Adresse adressiert, die mit der `ESTABLISHED`-Nachricht vom nächsten Gateway (rechts) eingetroffen sind.

8. Die `ESTABLISHED`-Nachricht wird mit der eigenen Rewriting-Adresse an das vorherige Gateway gesendet.

9. Das Initiator-Gateway sendet eine `ESTABLISHED`-Nachricht mit einer Rewriting-Adresse, die als Verbindungsadresse dient, an die Quelle; auf dem Initiator-Gateway wird das Address-Rewriting wie in den Schritten 4 und 7 aktiviert, aber es wird zusätzlich noch die Absenderadresse durch die Rewriting-Adresse ersetzt, die das Initiator-Gateway mit der `CONNECT`-Nachricht übertragen hatte.

Die Senke ist am Verbindungsaufbau weder aktiv noch passiv beteiligt. Sie kann die bidirektionale Verbindung nutzen, indem sie auf Nachrichten, die von der Quelle über die Verbindung eintreffen, antwortet. Die Teilnahme der Senke am Verbindungsaufbau ist unnötig und würde diesen komplizierter machen, insbesondere in der Praxis, da die Senke dann einen Pfadschaltungsdienst betreiben müsste. Falls die Senke die Verbindungsadresse explizit erfahren soll, so kann das direkt durch die Quelle bzw. den Verbindungsauslöser geschehen, die Gateways sind davon nicht betroffen.

Abbildung 5.3 zeigt das zu dem Verbindungsaufbau in Abbildung 5.2 gehörende Address-Rewriting. Die Quelle hat für die Nutzung der Verbindung eine Verbindungsadresse erhalten. Die eigentliche Adresse der Senke könnte zwar auch als Verbindungsadresse verwendet werden, allerdings hat das

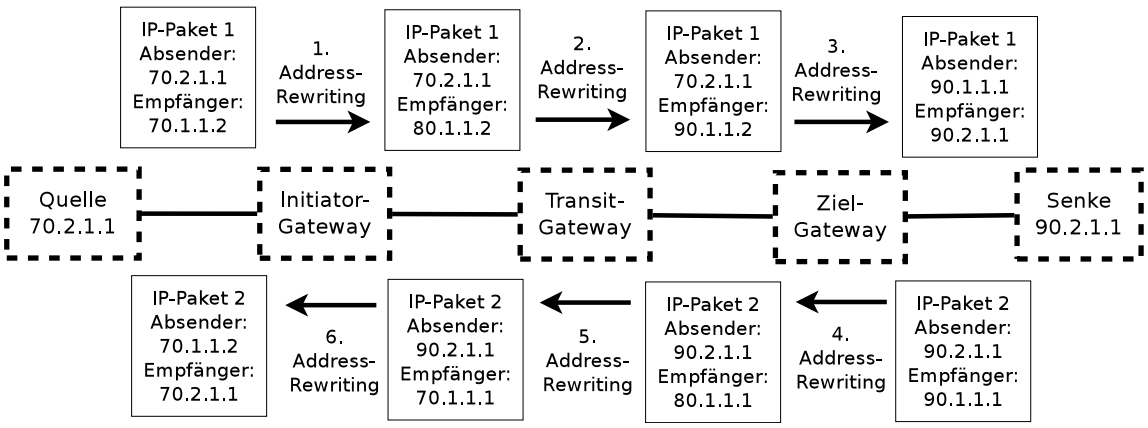


Abbildung 5.3.: Address-Rewriting für die in Abbildung 5.2 aufgebaute Verbindung

einige, wahrscheinlich unerwünschte Konsequenzen und Voraussetzungen: Einerseits kann die Senke nicht mehr anhand dieser Adresse über normales Routing erreicht werden (davon sind die Quelle und möglicherweise weitere Endpunkte betroffen), d.h. sämtlicher Verkehr läuft über die Verbindung. Andererseits sind individuelle Eingriffe in das Routing nötig, damit die Adresse der Senke über das Initiator-Gateway geroutet wird. Daher sprechen Quelle und Senke die Verbindung über ausgehandelte Verbindungsadressen an.

Die Rewriting-Vorgänge auf den Gateways sind:

1. IP-Paket 1, das von der Quelle an die Verbindungsadresse gesendet wurde, wird im Initiator-Gateway an die Rewriting-Adresse, die aus der `ESTABLISHED`-Nachricht von dem Transit-Gateway stammt, adressiert.
2. Das Transit-Gateway adressiert IP-Paket 1 an die Rewriting-Adresse, die aus der `ESTABLISHED`-Nachricht von dem Ziel-Gateway stammt.
3. Das Ziel-Gateway adressiert IP-Paket 1 an die Senke und ersetzt die Absenderadresse durch die Rewriting-Adresse, die das Transit-Gateway mit der `CONNECT`-Nachricht geschickt hatte. Das ist aus Sicht der Senke die Verbindungsadresse.
4. IP-Paket 2 ist die Antwort der „Senke“ (die jetzt die Rolle der Quelle übernimmt) auf IP-Paket 1 und wird im Ziel-Gateway an die aus der `CONNECT`-Nachricht vom Transit-Gateway stammende Rewriting-Adresse adressiert.
5. Das Transit-Gateway adressiert IP-Paket 2 an die Rewriting-Adresse, die aus der `CONNECT`-Nachricht vom Initiator-Gateway stammt.
6. Das Initiator-Gateway adressiert IP-Paket 2 an die „Quelle“ (die hier als Senke fungiert) und ersetzt die Absenderadresse durch die Verbindungsadresse.

Das Konzeptprotokoll kennt nur zwei Zustände, den Zustand `CONNECTING` und den Zustand `ESTABLISHED` (entspricht `KEEPALIVE`, vergleiche Abschnitt 4.3). Der Zustand `CONNECTING` wird durch das Eintreffen der `CONNECT`-Nachricht ausgelöst, der Zustand `ESTABLISHED` entsprechend durch die `ESTABLISHED`-Nachricht. Im Zustand `CONNECTING` werden bereits alle Ressourcen inklusive einer Rewriting-Adresse reserviert, damit bei Eintreffen der Verbindungsbestätigung die Ressourcen auch verfügbar sind und die Verbindung nicht mehr scheitern kann. Der Verbindungsabbau geschieht durch eine `DISCONNECT`-Nachricht, welche wie die `CONNECT`-Nachricht weitergeleitet wird. Nach dem Eintreffen der `DISCONNECT`-Nachricht werden (idealisiert) gleichzeitig das Rewriting deaktiviert, die Ressourcen freigegeben und eine `DISCONNECT`-Nachricht an das nächste Gateway gesendet. Durch die Gleichzeitigkeit wird ein Zustand `CLOSING` nicht benötigt, mit dem Absenden der `DISCONNECT`-Nachricht kann die Verbindung gelöscht werden.

Mit dem einfachen Konzeptprotokoll ist schon viel möglich. Wenn man die Idealisierung aufgibt,

dass das beste nächste Gateway bereits bekannt ist, und stattdessen mehrere Gateways in Frage kommen, so kann mit diesem Protokoll schon eine Breitensuche durchgeführt werden. Das Initiator-Gateway schickt dazu an alle in Frage kommenden Gateways eine `CONNECT`-Nachricht und wartet, welches Gateway zuerst mit einer `ESTABLISHED`-Nachricht antwortet. Allen anderen Gateways wird eine `DISCONNECT`-Nachricht gesendet. Das impliziert allerdings die Idealisierung, dass keine Überschneidungen bei der Suche aufgetreten sind, andernfalls kann die `DISCONNECT`-Nachricht auch einen Teil der Verbindung beenden, die nicht abgebaut werden soll (siehe auch Abschnitt 4.3.2).

Die Idealisierung ist für folgende Einschränkungen des Konzeptprotokolls verantwortlich:

- Verlust der Pfadschaltungsnachrichten wird nicht bemerkt
- Verbindungsaufbau kann stillstehen (durch Nachrichtenverlust)
- Verwaiste Verbindung und Ressourcen (Fehler werden nicht erkannt und kommuniziert)
- Keine funktionierende Pfadsuche
- Verbindungen können nicht identifiziert und Überschneidungen nicht erkannt werden
- Ressourcenengpässe und Reaktionen darauf sind nicht berücksichtigt
- Pfadänderungen sind nicht möglich
- Keine Authentifizierung und Autorisierung von Pfadschaltungsnachrichten
- Keine Fehlernachrichten und keine Fehlerauswertung
- Undefinierter Verbindungszustand nach Ausfall eines Gateways

Die Idealisierung wird hier aufgegeben. Im nächsten Abschnitt werden weitere Protokollnachrichten und Zustände eingeführt, die diese Einschränkungen aufheben und das Konzeptprotokoll zum Path-Parameter-Control-Protocol ausbauen, welches die Pfadschaltung, die in Kapitel 4 konzipiert wurde, umsetzt.

5.2. PPCP-Nachrichten und Kommunikationsmuster

Die Nachrichten, die dem Konzeptprotokoll fehlen, ergeben sich aus Kapitel 4, insbesondere Abschnitt 4.3, Abbildung 4.5 und Abschnitt 4.6. Es werden Nachrichten zur Pfadsuche und zur Auswertung der Pfadsuche benötigt, Nachrichten zur Pfadschaltung und für den Verbindungsabbau sind bereits im Konzeptprotokoll enthalten und werden hier übernommen und nach Bedarf erweitert. Um verwaiste Verbindungen und blockierte Ressourcen zu vermeiden, werden Keepalive-Nachrichten eingeführt und der Zustand `ESTABLISHED` aus dem Konzeptprotokoll wird entsprechend durch den Zustand `KEEPALIVE` ersetzt. Wie in Abschnitt 4.6.1 dargelegt wurde, werden Quittungsnachrichten für jeden Nachrichtentyp benötigt. Dort werden außerdem Fehler- und Umleitungsnachrichten gefordert. Um eine gewisse Konsistenz in der Nachrichtensemantik zu wahren, sollen alle Quittungsnachrichten nur Informationen darüber enthalten, welche Nachricht quittiert wurde.

Die angesprochenen Nachrichten werden in drei Übersichtsbildern vorgestellt, je eines mit Kommunikationsmustern für den Verbindungsaufbau, die Verbindungserhaltung und den Verbindungsabbau. Abbildung 5.4 zeigt die Nachrichten, die am Verbindungsaufbau beteiligt sind. In der Abbildung ist auch der Fall berücksichtigt, dass ein Ende (der Verbindungsauslöser) einige Nachrichten des Path-Parameter-Control-Protocol senden kann (z. B. SEARCH und CONNECT). Die Auslösung der Verbindung kann aber auch durch einen Verbindungsauslösungsdienst durchgeführt werden (siehe Abschnitt 4.2). Den Nachrichten wird eine Richtung zugeordnet, „→“ bedeutet von Quelle in Richtung Senke, „←“ bedeutet entsprechend die Gegenrichtung. Die Pfeile im Text beziehen sich auf die Richtung, in die die Nachricht gesendet werden kann. Nachrichten, die in beide Richtungen möglich sind (↔), werden in den konkreten Fällen auf den Abbildungen nur in eine Richtung gesendet. Die Vorgänge der Abbildung in Abbildung 5.4 sind im Einzelnen:

1. Die Pfadsuche auslösen (SEARCH →)

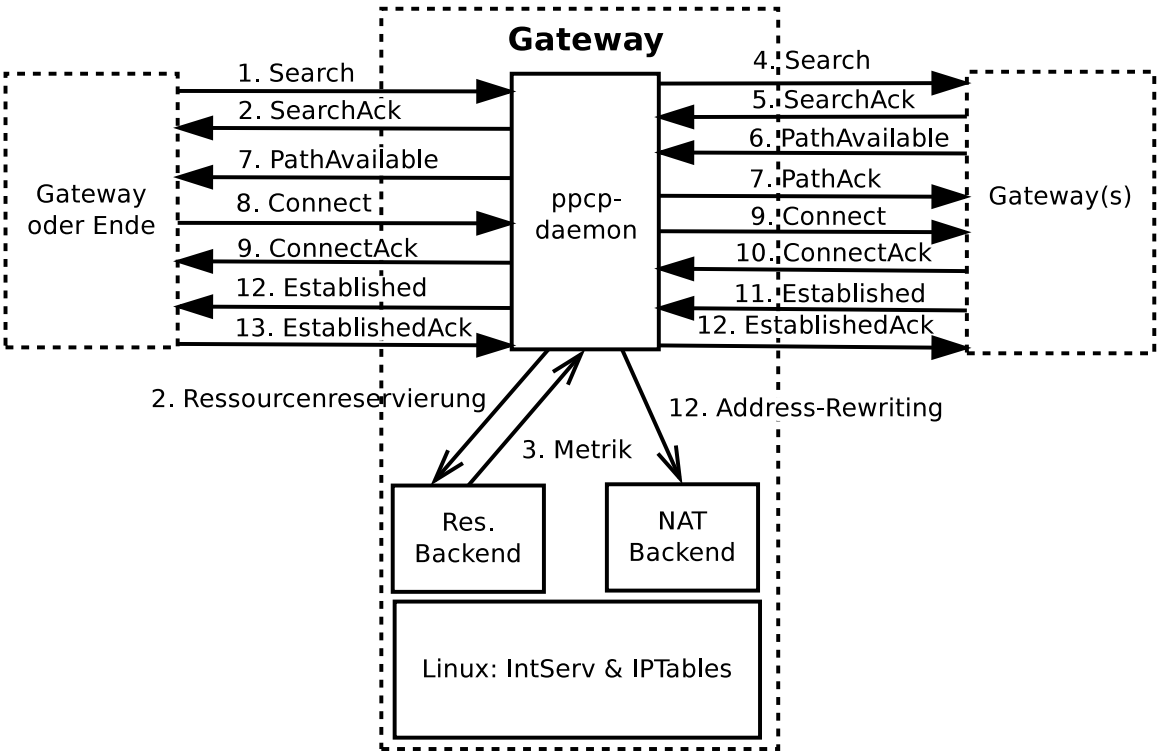


Abbildung 5.4.: Übersicht über die PPCP-Nachrichten für den Verbindungsaufbau

2. Die Pfadsuche quittieren (← SEARCHACK) und Ressourcen reservieren oder vormerken (siehe Abschnitt 4.4) und eine Metrik berechnen lassen
3. Auswertung einer Metrik, die die vorhandenen Ressourcen in ein Verhältnis zu den QoS-Anforderungen setzt (siehe Abschnitt 4.4.3)
4. Die Pfadsuche wird an ein oder mehrere nächste Gateways auf dem Pfad weitergeleitet (SEARCH →)
5. Die Quittung(en) für die Pfadsuche empfangen (← SEARCHACK)
6. Ein oder mehrere Gateways haben einen oder mehrere Pfade gefunden (← PATHAVAILABLE)
7. Das Suchergebnisse quittieren (← PATHACK) und Suchergebnis weiterleiten (← PATHAVAILABLE)
8. Die Pfadschaltung auslösen (CONNECT →)
9. Die Pfadschaltung quittieren (← CONNECTACK) und weiterleiten (CONNECT →)
10. Quittung für die Pfadschaltung empfangen (← CONNECTACK)
11. Die Pfadschaltung wurde auf den nächsten Gateways durchgeführt (← ESTABLISHED)
12. Die Pfadschaltung lokal durchführen (Address-Rewriting), die Durchführung quittieren (ESTABLISHEDACK →) und weiterleiten (← ESTABLISHED)
13. Quittung der Durchführung empfangen (ESTABLISHEDACK →)

Aus Sicht des Gateways ist der Pfad geschaltet und die Verbindung damit aufgebaut. Die Keepalive-Nachrichten dienen der Aufrechterhaltung der Verbindung. Die Keepalive-Nachrichten werden ebenfalls quittiert. Um die Quittungen einer Keepalive-Nachricht zuordnen zu können, enthält die Keepalive-Nachricht einen Zeitstempel, der von der Quittung wiederholt wird. Ohne die Zeitstempel kann es dazu kommen, dass eine neue Keepalive-Nachricht durch alte Quittungen bestätigt würde und so eine

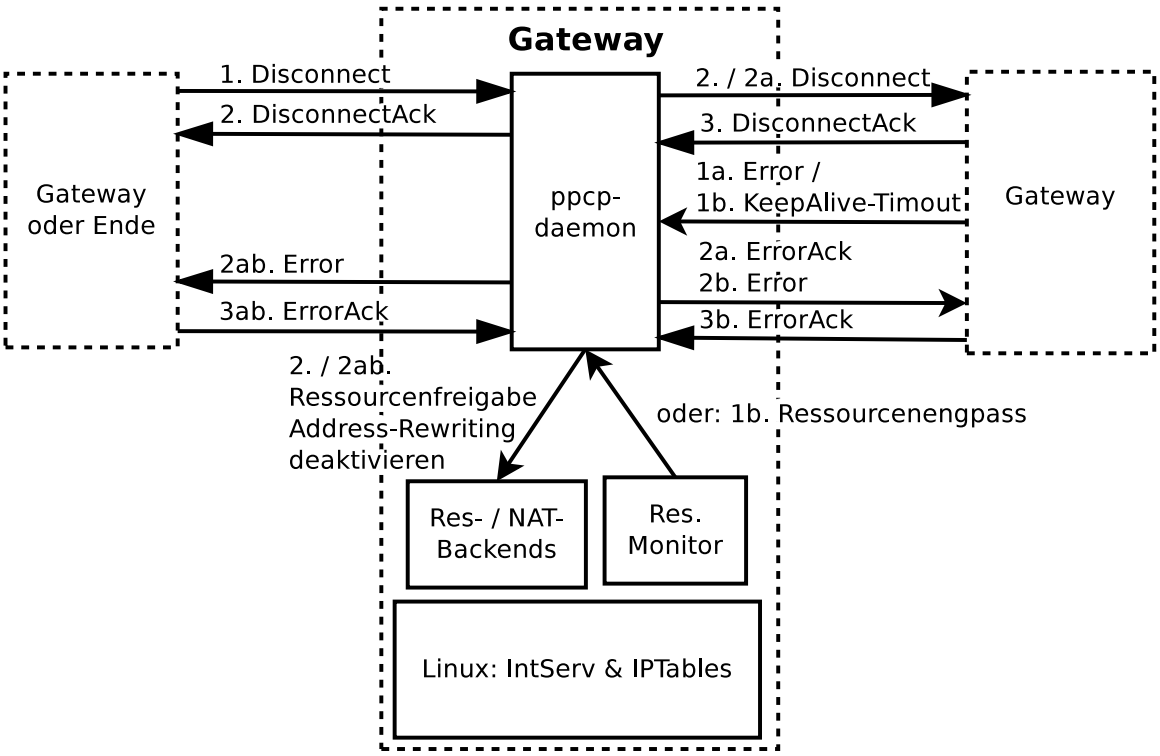


Abbildung 5.5.: Übersicht über die PPCP-Nachrichten, die den Verbindungsabbau auslösen und steuern können

Verbindung aufrecht erhalten wird, die eigentlich schon unterbrochen ist (z. B. durch einen Replay-Angriff, mehr dazu in Abschnitt 5.3.3). Die Keepalive-Nachrichten und Quittungen bestätigen nur, dass die benachbarten Gateways nicht ausgefallen sind, die Ressourcen im eigenen AS müssen ebenfalls überwacht werden. Die interne Überwachung dieser Ressourcen wird aber nicht über das Path-Parameter-Control-Protocol abgewickelt, sondern geschieht durch die QoS-Backends. Die Benachrichtigung der Nachbar-Gateways über den Ausfall von Ressourcen wird wieder durch PPCP-Nachrichten vollzogen.

Der Verbindungsabbau kann durch unterschiedliche Ereignisse ausgelöst werden, einerseits durch den Verbindungsnutzer, andererseits durch Fehler oder ausbleibende Keepalive-Quittungen. Abbildung 5.5 gibt eine Übersicht über die PPCP-Nachrichten, die am Verbindungsabbau beteiligt sind:

1. Die Verbindung wird „von links“ beendet (DISCONNECT $\rightarrow$), ob durch einen Fehler oder durch den Verbindungsnutzer ist hier nicht sichtbar.
2. Der Empfang von DISCONNECT wird quittiert (DISCONNECT $\leftarrow$), die Ressourcen werden freigegeben, das Address-Rewriting deaktiviert und DISCONNECT weitergeleitet.
3. Das nächste Gateway quittiert die DISCONNECT-Nachricht.

1a. Das nächste Gateway meldet einen Fehler (ERROR $\leftrightarrow$), der in diesem Fall zum Abbau der Verbindung auf diesem Gateway führt (in anderen Fällen könnte die Entscheidung dem vorherigen Gateway überlassen werden).

2a. Ressourcen werden freigegeben, der Empfang der ERROR-Nachricht quittiert (ERRORACK $\leftrightarrow$) und eine ERROR-Nachricht an das vorherige Gateway gesendet; die Fehlernachricht kann Informationen über die Reaktion (DISCONNECT) des aktuellen Gateways beinhalten.

3a. Das vorherige Gateway bestätigt den Empfang (ERRORACK $\leftrightarrow$).

1b. Ein Ressourcenengpass oder Keepalive-Timeout löst einen Fehler aus.

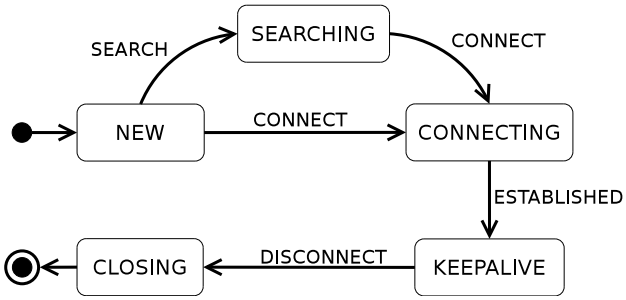


Abbildung 5.6.: Zustände einer Verbindung aus Gateway-Sicht

2b. Hier werden eine Fehlermeldung (ERROR ↔) an beide Nachbarn gesendet und Ressourcen freigegeben (in der Datenbank, real sind sie bereits ausgefallen), das Rewriting wird erst mit DISCONNECT deaktiviert; das weitere Vorgehen (SEARCH im Fall eines Keepalive-Timeouts oder DISCONNECT) ist nicht mehr abgebildet.

3b. Die Quittungen für die Fehlermeldungen werden empfangen (ERRORACK ↔).

Die folgenden notwendigen Nachrichten wurden in den Bildern nicht berücksichtigt:

- Verbindungsumleitung (REDIRECT ↔)
- Quittung für die Umleitungsnachricht (REDIRECTACK ↔)
- Suche abbrechen (SEARCHCANCEL →)
- Abbruch quittieren (← CANCELACK)
- Pfadsuche gescheitert (← SEARCHFAIL)
- Verbindungsaufbau gescheitert (← CONNECTFAIL)
- Quittung einer Scheiternachricht (FAILACK →)

Das Zustandsdiagramm aus Abschnitt 4.3 wird mit den PPCP-Nachrichten zu Abbildung 5.6, dabei wurden nur die Nachrichten einer erfolgreichen Verbindungsnutzung berücksichtigt. SEARCHFAIL, CONNECTFAIL und, wenn der Fehler nicht behoben werden kann, auch ERROR führen in den Zustand `CLOSING`. Wie auf dem Diagramm zu sehen ist, erlaubt auch das PPCP die Auslassung des Zustands `SEARCHING`.

In den folgenden Abschnitten werden Inhalt und Semantik aller PPCP-Nachrichten spezifiziert und nach Kommunkiationsvorgängen gruppiert.

5.2.1. Gemeinsame Elemente in allen Nachrichten

Wie schon in den Abschnitten 2.2.4 und 4.3.2 dargestellt, wird zur Identifikation einer Verbindung eine ID benötigt, Abschnitt 2.2.4 fordert zusätzlich noch einen Verbindungsbesitzer, der die ID vergibt und lokal deren Eindeutigkeit garantieren kann. Der Besitzer ist immer ein Gateway und wird durch seine Adresse identifiziert. Für jede Nachricht ist die Angabe ihres Typs obligatorisch, ein Zeitstempel ist optional und kann in allen Nachrichten verwendet werden, beispielsweise um alte Nachrichten zu erkennen. Weiterhin hat jede Nachricht einen Empfänger und einen Absender.

Der Besitzer und die ID sind global gültig, Empfänger und Absender hängen dagegen von der Kante des Pfades ab, auf der sie übertragen werden. Das Feld Nachrichtentyp gilt dann als global, wenn die Nachricht vom Initiator-Gateway ausgelöst wurde. Das bedeutet nicht, dass sie auch vom Initiator-Gateway stammt. Die `SEARCH`-Nachricht wird beispielsweise zu Beginn vom Initiator-Gateway ausgelöst und in jedem beteiligten Gateway verändert weitergeleitet. Dagegen sind die `ERROR`-Nachrichten

in der Regel nicht global. Abschnitt 2.2.4 schlägt eine Signatur vor, um die Parteien der Pfadschaltung zu authentifizieren. Dazu wird in den Nachrichten ein Signatur-Feld vorgesehen, das sowohl die Signatur der globalen Elemente als auch ein Feld, das die Signatur für die komplette Nachricht (inklusive der globalen Signatur) enthält (siehe Abschnitt 5.3.3).

Ein Nachrichtenskelett, das als Vorlage für die Nachrichten der folgenden Abschnitte dient, enthält also folgende Felder:

1. Initiator-Gateway (der globale Verbindungsbesitzer)
2. Zeitstempel des Initiator-Gateways (optional)
3. Verbindungs-ID (global)
4. Signatur des Initiator-Gateway (für globale Felder, optional)
5. Nachrichtentyp (kann global und lokal sein)
6. Empfänger (lokal)
7. Absender (lokal)
8. Zeitstempel des Absenders (lokal, nicht immer optional)
9. Signatur des Absenders (optional)

5.2.2. PPCP-Nachrichten für die Pfadsuche: `SEARCH` und `PATHAVAILABLE`

Aus Sicht des PPCP beginnt ein Verbindungsaufbau mit dem Anstoßen der Pfadsuche durch das Initiator-Gateway (die Kommunikation zwischen Verbindungsnutzer und Initiator-Gateway wird unabhängig realisiert, siehe Konzeption in Abschnitt 4.2). In Abschnitt 4.3.2 und Abschnitt 4.4.3 wurde eine Pfadsuche konzipiert. Die `SEARCH`-Nachricht und ihre assoziierten Nachrichten sind in Abbildung 5.7 zu sehen. Initiator-Gateway und Verbindungs-ID identifizieren die Nachricht, der Zeitstempel des Initiator-Gateway ist optional und dient dazu, veraltete Nachrichten zu erkennen. Falls die Transit-Gateways die Möglichkeit haben sollen, einen Teil der Verbindung umzuleiten, so müssen sie auch später `SEARCH`-Nachrichten für die Suche einer Umleitung versenden (siehe Abschnitt 5.3.3). Die Signatur des Initiator-Gateways ist dagegen auch später nutzbar, solange der Zeitstempel des Initiator-Gateways nicht genutzt wurde (oder er andernfalls noch aktuell genug ist).

Der Zeitstempel des Absenders hat nicht nur den Zweck, sein Alter anzugeben, sondern dient auch zur Identifikation der quittierten Nachrichten in den Quittungsnachrichten. Die ID genügt nicht, da sie nur die Verbindung, aber nicht einzelne Nachrichten identifiziert. Da sich die Quittungsnachrichten immer auf eine bestimmte (über ihren Zeitstempel identifizierbare) Nachricht beziehen, benötigen sie keinen eigenen Zeitstempel. Treffen sie zu einem unerwarteten Zeitpunkt ein, so können sie verworfen werden. Wie im Abschnitt über das Konzeptprotokoll zu sehen ist, wird bei einer bidirektionalen Verbindung im Ziel-Gateway die Absenderadresse der Verbindungspakete durch die senkenseitige Verbindungsadresse ersetzt. Für eine bidirektionale Verbindung müssen daher beide Verbindungsendpunkte in den `SEARCH`-Nachrichten angegeben werden. Bei einer unidirektionalen Verbindung wird die Adresse der Quelle in den `SEARCH`-Nachrichten nicht benötigt (sie kann allerdings für Multiplexing verwendet werden, siehe nächster Absatz).

Die Verbindungsoptionen müssen nicht angegeben werden. Es wird davon ausgegangen, dass es sich um eine unidirektionale Verbindung ohne garantierte Ressourcen handelt. Die Semantik der Verbindungsoptionen „garantierte Ressourcen“ und „transparentes Multiplexing“ wurde in den Abschnitten 4.4.2 und 4.5.3 beschrieben. Hier sei noch angemerkt, dass die Option „transparentes Multiplexing“ von den Gateways gesetzt wird, die Multiplexing unterstützen und anbieten. Die Gateways, die eine `SEARCH`-Nachricht mit gesetzter Multiplexing-Option empfangen, können dann entscheiden, ob für mehrere Verbindungen dieselbe Rewriting-Adresse verwendet wird. Für das Demultiplexing wird allerdings auch eine Absenderadresse benötigt, auch wenn es sich um eine unidirektionale Verbindung handelt.

SEARCH
Initiator-Gateway
Verbindungs-ID
Zeitstempel des Initiator-Gateway (optional)
Signatur des Initiator-Gateway (optional)
Empfängeradresse
Absenderadresse
Zeitstempel des Absenders
Verbindungsendpunkt 1 bzw. Quelle (optional)
Verbindungsendpunkt 2 bzw. Senke
Verbindungsoptionen (optional): unidirektional (standard), bidirektional, garantierte Ressourcen, transparentes Multiplexing
Metrik
QoS-Parameter (optional): minimale Übertragungsrate, maximale Latenz, maximaler Jitter, maximale Paketverlustrate, QoS-ID zur Identifikation zuvor definierter Anforderungsmengen, eingebettete QoS-Parameter
Signatur des Absenders (optional)

SEARCHFAIL
Initiator-Gateway
Verbindungs-ID
Empfängeradresse
Absenderadresse
Zeitstempel des Absenders
Signatur des Absenders (optional)

SEARCHACK / FAILACK
Initiator-Gateway
Verbindungs-ID
Empfängeradresse
Absenderadresse
Zeitstempel der bestätigten Nachricht
Signatur des Absenders (optional)

Abbildung 5.7.: Die PPCP-Nachrichten SEARCH, SEARCHFAIL, SEARCHACK und FAILACK

Die SEARCH-Nachrichten verwenden die in Abschnitt 4.4.3 definierte Metrik. Solange nur die vier Standard-QoS-Parameter verwendet werden, hat die Metrik die dort beschriebene Semantik, gibt also, informell gesagt, den „Verbrauch“ der QoS-Parameter in Prozent an und dient damit gleichzeitig als Time-to-live-Mechanismus. Dafür wird implizit als fünfter QoS-Parameter die Anzahl der beteiligten Gateways eingeführt. Jedes Gateway muss die Metrik mindestens um 1 reduzieren (für maximal 100 beteiligte Gateways). Werden QoS-IDs oder eingebettete Parameter verwendet, so kann die Metrik im Rahmen der Definition dieser Parameter ebenfalls neu definiert werden, sodass die Verwendung der Metrik nicht eingeschränkt ist.

Die QoS-Parameter kommen hier, wie in Abschnitt 4.4.2 definiert, zum Einsatz. Die SEARCH-Nachricht besagt, dass die vom Empfänger der Nachricht initiierte Pfadsuche auf dem Absender fehlgeschlagen ist. Ausser der Identifikation der Verbindung und der Kommunikationspartner werden keine weiteren Informationen benötigt.

Die Signatur des Nachrichtenabsenders ist optional und wird in Abschnitt 5.3.3 beschrieben.

Abbildung 5.8 zeigt die Kommunikationsmuster von SEARCH:

1. Pfadsuche auslösen (SEARCH)
2. Pfadsuche quittieren (SEARCHACK) und die Senke bzw. die nächsten Gateways suchen; falls die Senke (der gesuchte Verbindungsendpunkt) im Zuständigkeitsbereich des Gateway liegt, geht es weiter auf Abbildung 5.10, anderenfalls werden die lokalen Pfadabschnitte zu den nächsten Gateways zusammen mit den QoS-Parametern an das Resource-Backend übergeben und eine Metrik berechnet; wurden keine geeigneten Gateways gefunden, springe zu Schritt 7.
3. Die Metrik wird ausgewertet. Ist sie negativ, springe zu 7, ist sie positiv, werden die Ressourcen für die Verbindung vorgemerkt.
4. Die Pfadsuche wird bei den nächsten Gateways ausgelöst (SEARCH).
5. Quittung der Pfadsuche empfangen (SEARCHACK), wenn kein Fehler auftritt, geht es weiter auf Abbildung 5.10.

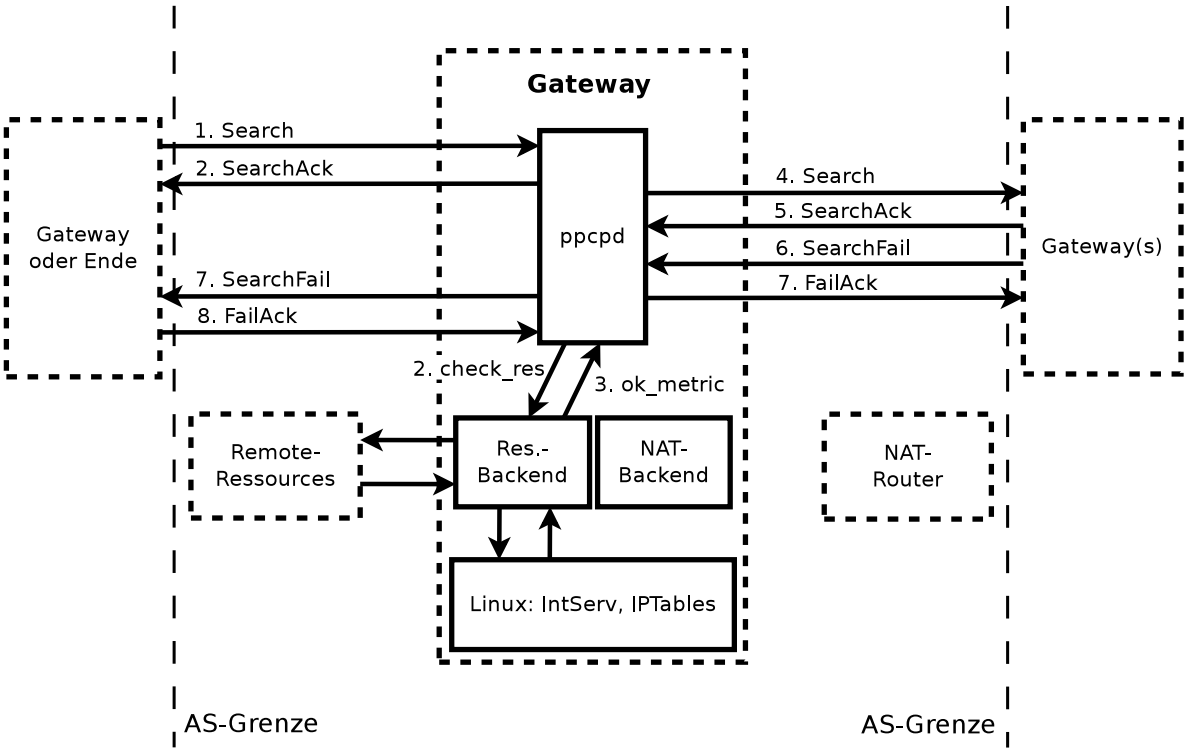


Abbildung 5.8.: Kommunikationsmuster von `SEARCH`

6. `SEARCHFAIL` wird empfangen, wenn keine von diesem Gateway in Schritt 4 angestoßene Suche mehr aktiv ist, ist die Suche gescheitert.

7. Scheitern der Suche quittieren (`FAILACK`), vorgemerkte Ressourcen freigeben und `SEARCHFAIL` an das vorherige Gateway senden.

8. Das vorherige Gateway quittiert die gescheiterte Suche (`FAILACK`).

Sobald ein Gateway feststellt, dass die Senke bzw. der gesuchte Verbindungsendpunkt im eigenen Zuständigkeitsbereich liegt, wird die Metrik für den ganzen Pfad berechnet. Ist sie nicht negativ, so wurde ein Pfad gefunden. Die Metrik für den kompletten Pfad wird in der `PATHAVAILABLE`-Nachricht den vorherigen Gateways mitgeteilt. In Abbildung 5.9 ist die `PATHAVAILABLE`-Nachricht und ihre Quittung abgebildet. Die Quittung hat dieselben Felder und dieselbe Funktionsweise wie die Quittungen für `SEARCH` und `SEARCHFAIL`. Die Felder der `PATHAVAILABLE`-Nachricht sind ebenfalls bekannt, allerdings hat die Metrik hier eine andere Bedeutung als in der `SEARCH`-Nachricht: Sie gibt zwar immer

PATHAVAILABLE
Initiator-Gateway
Verbindungs-ID
Empfängeradresse
Absenderadresse
Zeitstempel des Absenders
Metrik
Signatur des Absenders (optional)

PATHACK
Initiator-Gateway
Verbindungs-ID
Empfängeradresse
Absenderadresse
Zeitstempel der bestätigten Nachricht
Signatur des Absenders (optional)

Abbildung 5.9.: Die PPCP-Nachrichten `PATHAVAILABLE` und `PATHACK`

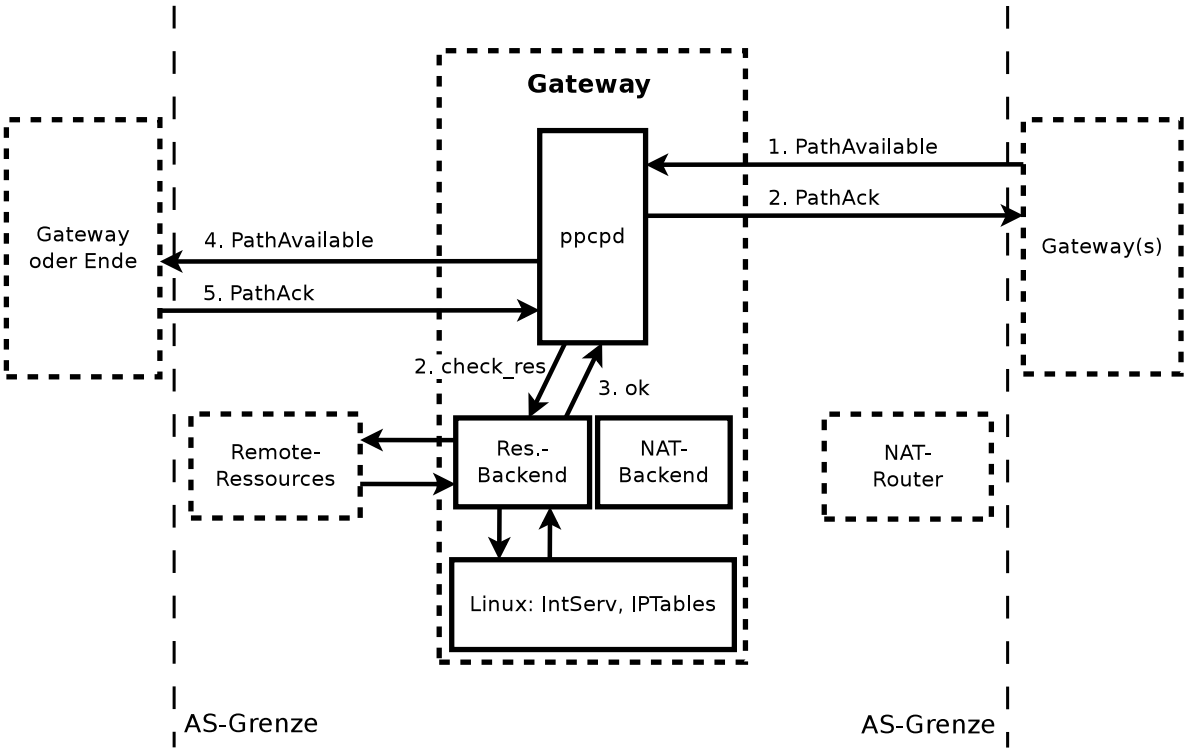


Abbildung 5.10.: Kommunikationsmuster von `PATHAVAILABLE`

noch den „Verbrauch“ der QoS-Parameter in Prozent an, allerdings bezieht sich diese Aussage auf den ganzen Pfad und impliziert eine erfolgreiche Pfadsuche. Es wäre auch möglich, negative Metriken über `PATHAVAILABLE` anzubieten, um dem Initiator-Gateway die Möglichkeit zu geben, unter den schlechten Pfaden den besten auszuwählen. Das Gleiche lässt sich aber auch durch eine weitere Suche mit bescheideneren QoS-Parametern erreichen. Daher gilt die Suche bei einer negativen Metrik als gescheitert und wird mit einer `SEARCHFAIL`-Nachricht abgebrochen.

Die Kommunikationsmuster von `PATHAVAILABLE` sind in Abbildung 5.10 zu sehen:

1. Ein Ergebnis der (erfolgreichen) Pfadsuche wird zusammen mit einer Metrik empfangen (`PATHAVAILABLE`).
2. Das Suchergebnis wird quittiert (`PATHACK`) und vor der Weiterleitung überprüft, ob die vorgemerkten Ressourcen anderweitig vergeben wurden; die Überprüfung der Ressourcen kann an dieser Stelle zum Scheitern der Pfadsuche führen, ist aber optional, da die Ressourcen erst bei der Pfadschaltung reserviert werden und sich die Ressourcenbelegung bis dahin wieder ändern kann.
3. Die Ressourcenbelegung wird ausgewertet.
4. Das Suchergebnis wird weitergeleitet.
5. Die Quittung für das Suchergebnis wird empfangen.

Erreichen ein oder mehrere Suchergebnisse das Initiator-Gateway, kann die Verbindung in den Zustand `CONNECTING` wechseln.

5.2.3. PPCP-Nachrichten für die Pfadschaltung: `CONNECT` und `ESTABLISHED`

Je nach verwendetem Suchalgorithmus und der Anzahl der für die Verbindung verfügbaren Nachbar-Gateways, erhält das Initiator-Gateway ein oder mehrere Suchergebnisse. Bei der Konzeption von

CONNECT
Initiator-Gateway
Verbindungs-ID
Zeitstempel des Initiator-Gateway (optional)
Signatur des Initiator-Gateway (optional)
Empfängeradresse
Absenderadresse
Zeitstempel des Absenders
Rewriting-Address (nur für bidirektionale Verbindungen)
Keepalive-Timeout
Metrik (ab hier bis einschließlich QoS-Parameter nur für die Suche „on the fly“)
Verbindungsendpunkt 1 bzw. Quelle
Verbindungsendpunkt 2 bzw. Senke
Verbindungsoptionen : wie bei SEARCH
QoS-Parameter: wie bei SEARCH
Signatur des Absenders (optional)

CONNECTFAIL
Initiator-Gateway
Verbindungs-ID
Empfängeradresse
Absenderadresse
Fehler-Code / Fehlernachricht (optional, siehe ERROR)
Zeitstempel des Absenders
Signatur des Absenders (optional)

CONNECTACK / FAILACK
Initiator-Gateway
Verbindungs-ID
Empfängeradresse
Absenderadresse
Zeitstempel der bestätigten Nachricht
Signatur des Absenders (optional)

Abbildung 5.11.: Die PPCP-Nachrichten CONNECT, CONNECTFAIL, CONNECTACK und FAILACK

PPCP wird davon ausgegangen, dass jedes Gateway nur ein Ergebnis weiterleitet. Das Protokoll kann zwar auch dazu verwendet werden, mehrere Ergebnisse weiterzuleiten, dieser Fall wird hier aber nicht behandelt. Die Pfadschaltung kann mit oder ohne vorherige Suche begonnen werden. Eine Pfadschaltung ohne vorherige Suche sucht den Pfad „on the fly“ und ist sinnvoll, wenn eine Tiefensuche gute Aussicht auf Erfolg hat. Auf die Konzepte der Pfadschaltung wird hier aber nicht näher eingegangen, sie sind in Abschnitt 4.3.3 nachzulesen.

Wird die Pfadsuche „on-the-fly“ durchgeführt, so enthält die CONNECT-Nachricht alle Felder der SEARCH-Nachricht (siehe Abbildung 5.11). Wurde eine Pfadsuche durchgeführt, so sollten die Felder Metrik, Verbindungsendpunkt 1 und 2, Verbindungsoptionen und QoS-Parameter in der CONNECT-Nachricht nicht verwendet und vom PPCP-Daemon ignoriert werden, da sich das Ergebnis der Pfadsuche auf die in den SEARCH-Nachrichten angegebenen Parameter bezieht.

Die CONNECT-Nachricht enthält zwei neue Felder, zum einen das Keepalive-Timeout, d. h. die maximale Zeitdifferenz zwischen zwei KEEPALIVE-Nachrichten (siehe Abschnitt 5.2.4) und zum anderen eine Rewriting-Adresse für bidirektionale Verbindungen. Letzteres Feld wird nur in Kombination mit der entsprechenden Verbindungsoption verwendet.

Die CONNECTFAIL-Nachricht ist wie die SEARCHFAIL-Nachricht aufgebaut, sieht aber noch die Definition eines Fehler-Codes vor, der beispielsweise dazu verwendet werden kann, das Gateway-Lookup bei der nächsten Suche zu beeinflussen. Die Definition der Fehler-Codes ist aber nicht Teil von PPCP, die CONNECTFAIL-Nachricht dient lediglich als Container. CONNECTACK und FAILACK sind wie alle Quittungsnachrichten aufgebaut.

Die Kommunikationsmuster von CONNECT und die assoziierten Vorgänge auf den Gateways sind in Abbildung 5.12 zu sehen:

1. Die Aufforderung zur Schaltung eines Pfades trifft ein (CONNECT).
2. Der Empfang wird quittiert (CONNECTACK) und das nächste Gateway ausgewählt (entweder anhand von Suchergebnissen oder „on-the-fly“ per Gateway-Lookup) und Ressourcen sowie eine Rewriting-Adresse werden angefordert; bei einer bidirektionalen Verbindung wird eine weitere

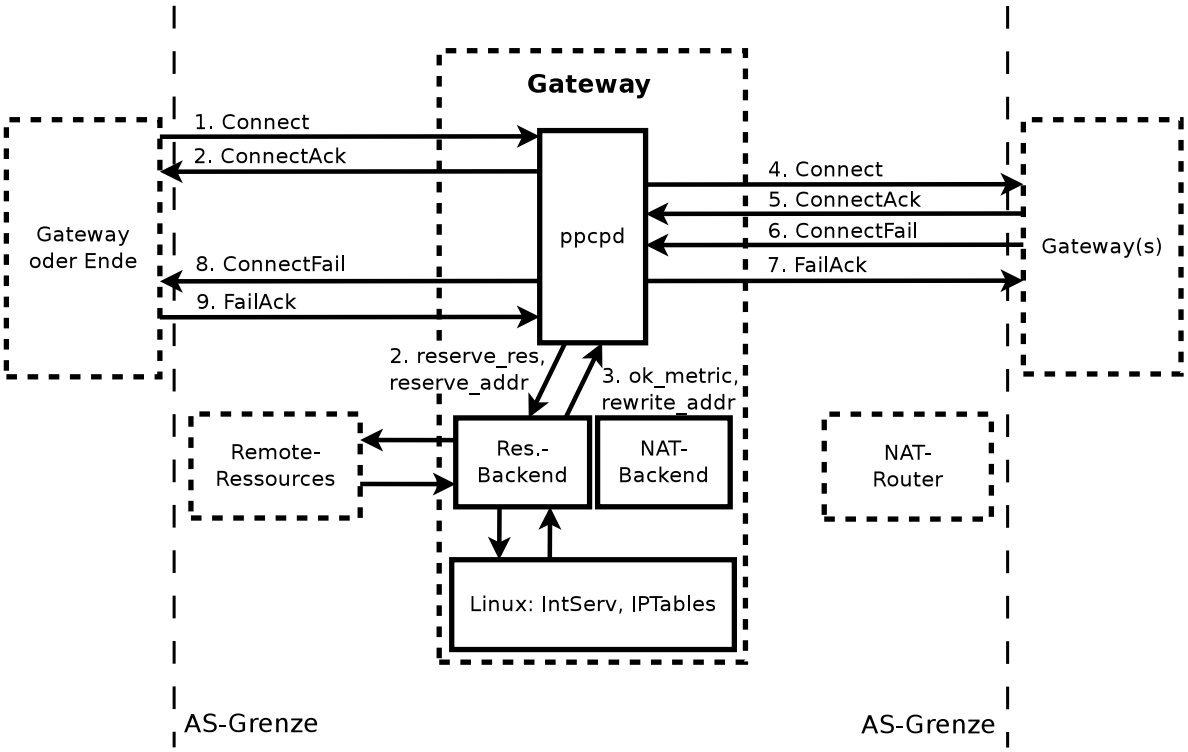


Abbildung 5.12.: Kommunikationsmuster von `CONNECT`

Rewriting-Adresse angefordert.

3. Die Metrik für den lokalen Ressourcenverbrauch wird berechnet; wurde zuvor eine Pfadsuche durchgeführt, so muss die aktuell berechnete Metrik mit der bei der Pfadsuche berechneten Metrik verglichen werden; ist das Ergebnis schlechter, so muss entschieden werden, ob die Pfadschaltung hier scheitert; lässt die Metrik, die mit `PATHAVAILABLE` als Suchergebnis eingetroffen ist, Spielraum, so kann die Pfadschaltung fortgesetzt werden, andernfalls müssen die Alternativen bewertet und eine Auswahl getroffen werden; Beispiele für Alternativen: Pfadschaltung fortsetzen, anderen Zweig versuchen, Pfadschaltung abbrechen oder Pfadschaltung umleiten (siehe `REDIRECT`-Nachricht im nächsten Abschnitt); wird die Pfadschaltung abgebrochen, springe zu Schritt 7.

4. War die Anforderung von Ressourcen erfolgreich, so wird die berechnete Metrik und eventuell eine Rewriting-Adresse (zur Aushandlung der Adressen siehe Abschnitt 5.1.1) an das zuvor ausgewählte Gateway gesendet (`CONNECT`).

5. Die Quittung der Pfadschaltungsnachricht wird empfangen (`CONNECTACK`); tritt bei der Pfadschaltung in den nächsten Gateways kein Fehler auf, so geht es mit Abbildung 5.13 weiter.

6. Das nächste Gateway meldet das Scheitern der Pfadschaltung (`CONNECTFAIL`).

7. Die Nachricht wird quittiert (`FAILACK`); hier kann zu Schritt 2 gesprungen und ein anderes Gateway ausgewählt werden („on-the-fly“ oder falls weitere Suchergebnisse verfügbar sind); kommt kein weiteres Gateway in Frage, geht es weiter mit Schritt 8.

8. Lokale Ressourcen und Adressen der Verbindung werden freigegeben und das Scheitern der Pfadschaltung dem vorherigen Gateway gemeldet (`CONNECTFAIL`).

9. Das vorherige Gateway quittiert den Empfang (`FAILACK`).

Das Gateway, in dessen Zuständigkeitsbereich sich die Senke befindet, berechnet nach dem Eintreffen der `CONNECT`-Nachricht die Metrik für den ganzen Pfad und stellt fest, ob die Pfadschaltung erfolgreich

ESTABLISHED
Initiator-Gateway
Verbindungs-ID
Empfängeradresse
Absenderadresse
Zeitstempel des Absenders
Rewriting-Address
Signatur des Absenders (optional)

SEARCHCANCEL
Initiator-Gateway
Verbindungs-ID
Empfängeradresse
Absenderadresse
Zeitstempel des Absenders
Signatur des Absenders (optional)

ESTABLISHEDACK / CANCELACK
Initiator-Gateway
Verbindungs-ID
Empfängeradresse
Absenderadresse
Zeitstempel der bestätigten Nachricht
Signatur des Absenders (optional)

Abbildung 5.13.: Die PPCP-Nachrichten ESTABLISHED, SEARCHCANCEL, ESTABLISHEDACK und CANCELACK

war. Das Scheitern wird mit der oben beschriebenen CONNECTFAIL-Nachricht mitgeteilt, der Erfolg mit der ESTABLISHED-Nachricht (Abbildung 5.13). Die Rewriting-Adresse, die beim Eintreffen von CONNECT reserviert wurde, wird jetzt dem vorherigen Gateway über die ESTABLISHED-Nachricht mitgeteilt. Wurde eine Breitensuche durchgeführt, so speichern mehrere Gateways Suchergebnisse und bleiben im Zustand SEARCHING, als bei der Pfadschaltung beteiligt sind. Nur die Gateways, die die CONNECT-Nachricht erhalten, wecheln in den Zustand CONNECTING. Spätestens mit dem Eintreffen der ESTABLISHED-Nachricht müssen die Suchergebnisse und Ressourcen in den anderen Zweigen mit der SEARCHCANCEL-Nachricht zur Löschung freigegeben werden. Auch beim Senden der SEARCHCANCEL-Nachricht kann es zu Überschneidungen kommen, und so kann ein Gateway, das an der Pfadschaltung beteiligt ist, diese Nachricht erhalten. Daher wird an dieser Stelle darauf hingewiesen, dass SEARCHCANCEL nur von Gateways im Zustand SEARCHING verarbeitet werden soll.

Die Kommunikationsmuster von ESTABLISHED und die assoziierten Vorgänge auf den Gateways sind in Abbildung 5.14 zu sehen:

1. Die erfolgreiche Pfadschaltung wird vom nächsten Gateway bestätigt und eine Rewriting-Adresse übermittelt (ESTABLISHED)

2. Quittierung der erfolgreichen Pfadschaltung (ESTABLISHEDACK); die reservierten Ressourcen werden der Verbindung zugeordnet; das ist in der Regel erst jetzt möglich, da dafür die Rewriting-Adresse benötigt wird (siehe Abschnitt 4.4.1); das Address-Rewriting wird aktiviert (siehe Abschnitt 5.1.1); falls noch Gateways im Zustand SEARCHING sind, bei welchen die Pfadsuche von diesem Gateway ausgelöst wurde, wird die Suche (spätestens jetzt) dort abgebrochen (SEARCHCANCEL); die zuvor reservierte Rewriting-Adresse wird zusammen mit der Bestätigung der Pfadschaltung an das vorherige Gateway weitergeleitet (ESTABLISHED)

3. Quittierung der Nachrichten (CANCELACK und ESTABLISHEDACK)

Die Verbindung steht, sobald das Initiator-Gateway die ESTABLISHED-Nachricht verarbeitet und den Verbindungsnutzer informiert hat und wechselt in den Zustand KEEPALIVE. Je nach Schnittstelle kann der Verbindungsnutzer beispielsweise ebenfalls mit einer ESTABLISHED-Nachricht benachrichtigt werden, die Rewriting-Adresse übernimmt hier die Rolle der Verbindungsadresse.

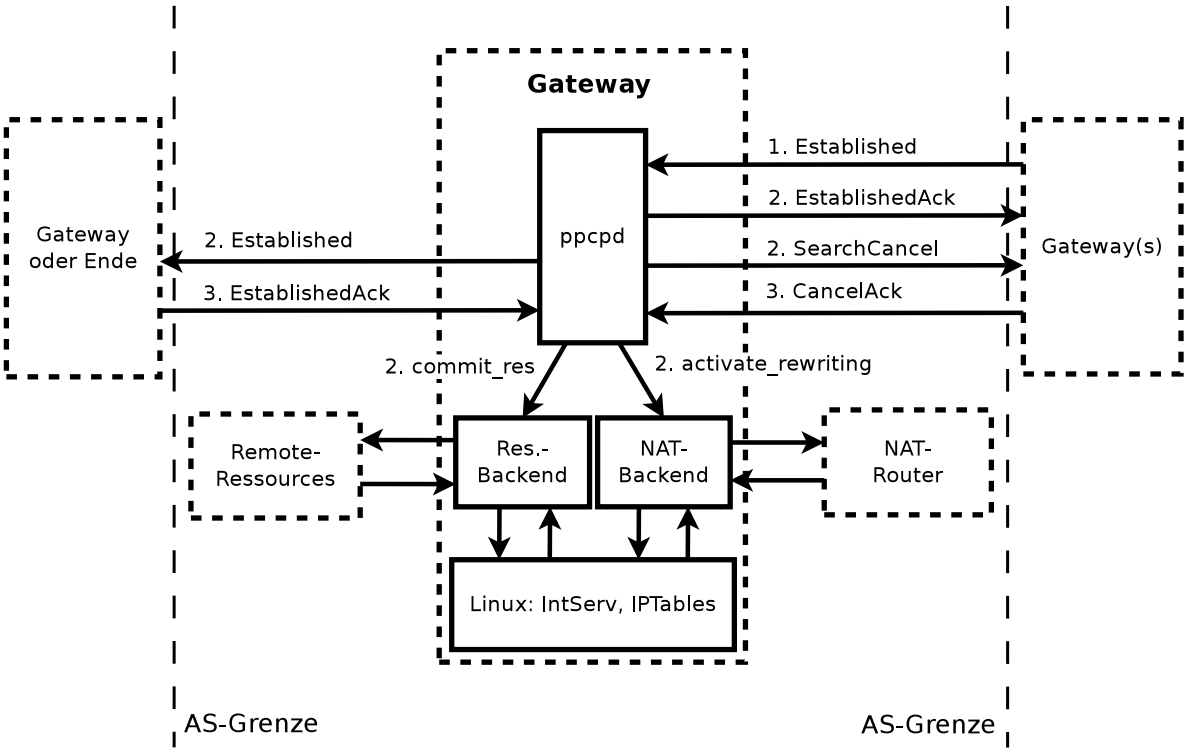


Abbildung 5.14.: Kommunikationsmuster von ESTABLISHED

5.2.4. PPCP-Nachrichten für Verbindungserhalt, die Fehlerbehandlung und den Verbindungsabbau: KEEPALIVE, ERROR, REDIRECT und DISCONNECT

Die KEEPALIVE-Nachrichten (Abbildung 5.15) dienen dazu, die Verbindung aufrechtzuerhalten und zu erkennen, ob ein Gateway ausgefallen ist. Wären Ausfälle und Fehler vollkommen ausgeschlossen, so wären die Keepalive-Nachrichten nicht nötig. Entsprechend den Anforderungen an die Verfügbarkeit und abhängig von der Verlässlichkeit, können KEEPALIVE-Nachrichten in kürzeren oder längeren Zeitintervallen gefordert werden. Dazu wird bei der Pfadschaltung (CONNECT) ein Keepalive-Timeout vereinbart, das je für zwei Gateways gilt. Werden in eingebetteten QoS-Parametern keine expliziten Anforderungen an Verfügbarkeit und Keepalive gemacht, so kann das Keepalive-Timeout von den Gateways bestimmt und später durch die KEEPALIVE-Nachrichten angepasst werden (Feld „Keepalive-Timeout“).

Wie auf Abbildung 5.16 zu sehen ist, wird die KEEPALIVE-Nachricht immer an das nächste Gate-

KEEPALIVE
Initiator-Gateway
Verbindungs-ID
Empfängeradresse
Absenderadresse
Zeitstempel des Absenders
Keepalive-Timeout (optional)
Signatur des Absenders (optional)

KEEPALIVEACK
Initiator-Gateway
Verbindungs-ID
Empfängeradresse
Absenderadresse
Zeitstempel der bestätigten Nachricht
Signatur des Absenders (optional)

Abbildung 5.15.: Die PPCP-Nachrichten KEEPALIVE und KEEPALIVEACK

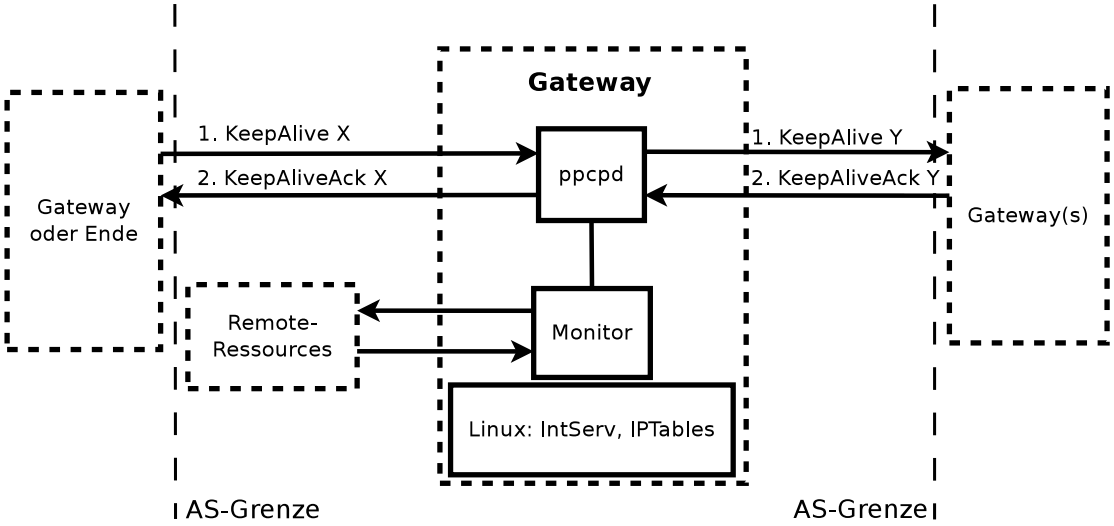


Abbildung 5.16.: Kommunikationsmuster von `KEEPALIVE`

way gesendet und das Timeout von dem Gateway bestimmt, das die `KEEPALIVE`-Nachrichten sendet. Die Identifikation von Verbindung und Nachricht wird hier wie gehabt durch Verbindungs-ID und Zeitstempel erreicht.

Trifft nach Ablauf des Keepalive-Timeouts keine Quittung ein, so kann die `KEEPALIVE`-Nachricht mehrmals hintereinander versendet werden, um die Wahrscheinlichkeit eines Nachrichtenverlusts zu verringern. Alternativ kann versucht werden, das nächste Gateway auf andere Weise (z. B. per TCP-Verbindung) zu erreichen. Auf welche Weise überprüft wird, ob ein Gateway ausgefallen ist, wird nicht im PPCP spezifiziert, sondern nur die Bedeutung des Keepalive-Timeouts: Verstreicht das Keepalive-Timeout einmal, so wird eine weitere `KEEPALIVE`-Nachricht versendet (möglicherweise mit einem kürzeren Keepalive-Timeout) und versucht, festzustellen, ob das Gateway ausgefallen ist. Nachdem das nächste (möglicherweise kürzere) Keepalive-Timeout abgelaufen ist, ohne dass neue Informationen gewonnen wurden, gilt das nächste Gateway als ausgefallen. Ist das Gateway nicht ausgefallen (z. B. festgestellt durch einen erfolgreichen TCP-Verbindungsaufbau), so gelten nur die `KEEPALIVE`-Nachrichten als verloren. In diesem Fall wird eine dritte `KEEPALIVE`-Nachricht versendet (nach Ermessen der Pfadschaltungsdienste beispielsweise über TCP). Wird auch die dritte Nachricht nicht rechtzeitig beantwortet, so gilt das Gateway auf jeden Fall als ausgefallen (auch, wenn ein TCP-Verbindungsaufbau erfolgreich war), und eine `ERROR`-Nachricht (Abbildung 5.17) wird an das vorherige Gateway gesendet. Ob gleichzeitig eine erneute Pfadssuche gestartet oder auf die Reaktion eines vorherigen Gateway (des Initiator-Gateway) gewartet wird, kann auf den Gateways eingestellt werden und ist nicht durch PPCP spezifiziert.

Bleibt nicht die Quittung sondern eine `KEEPALIVE`-Nachricht dreimal aus, so gilt das vorherige Gateway als ausgefallen und eine `ERROR`-Nachricht (Fehler-Code für Keepalive-Timeout) wird an die vorherige Nachricht gesendet. Nach obigem Muster wird die `ERROR`-Nachricht zwei- oder dreimal wiederholt (die Zeitabstände können vom Gateway gewählt werden). Bleibt eine Antwort aus, so wird die Verbindung von hier bis zur Senke abgebaut (`DISCONNECT`). Trifft in dieser Zeit eine `SEARCH`- oder `CONNECT`-Nachricht für diese Verbindung ein (weil eines der vorherigen Gateways versucht, das ausgefallene Gateway zu umgehen), so kann die Verbindung aufrechterhalten und weiterverwendet werden.

Prinzipiell kann der *Umgang mit Timeouts* immer auf diese Weise gelöst werden:

1. Erfolgt auf die erste Nachricht keine (rechtzeitige) Antwort, so wird eine zweite Nachricht gesendet und auf andere Weise (z. B. TCP-Verbindung) getestet, ob das betroffene Gateway ausgefallen ist.
2. Erfolgt auf die zweite Nachricht keine Antwort und konnte nicht auf andere Weise festgestellt

werden, ob das Gateway ausgefallen ist, so gilt das Gateway ab diesem Zeitpunkt als ausgefallen und es erfolgt keine dritte Wiederholung der Nachricht.

3. Wurde im vorherigen Schritt festgestellt, dass das Gateway nicht ausgefallen ist, wird die Nachricht ein drittes Mal versendet, falls verfügbar, über einen zuverlässigen Kommunikationskanal (TCP); wird auch die dritte Nachricht nicht bestätigt, so gilt das Gateway für diese Verbindung als ausgefallen.

In manchen Situationen kann die Vorgehensweise vereinfacht werden. Bleibt beispielsweise bei einer Breitensuche mit vielen Gateways eine einzelne Quittung aus, so kann sofort mit `SEARCHCANCEL` reagiert werden.

Bei einem kontrollierten Ausfall, d. h. bei einem Ausfall, bei dem die Gateways noch kommunizieren können, wie beispielsweise beim Ausfall eines QoS-Backends, kann das betroffene Gateway versuchen, die Verbindung mit einer `REDIRECT`-Nachricht (Abbildung 5.17) umzuleiten. Dazu wird dem vorherigen Gateway ein Ersatz-Gateway (Feld „Ersatz-Gateway“ in Abbildung 5.17) mitgeteilt, über das eine Pfadschaltung versucht wird, die vom vorherigen Gateway ausgeht. Ist diese Pfadschaltung erfolgreich, so kann das vorherige Gateway die Verbindung transparent (für die Verbindungsenden) umleiten, indem das Address-Rewriting auf die neue Rewriting-Adresse umgestellt wird. Der nicht mehr benötigte Pfadabschnitt kann dann kontrolliert deaktiviert werden.

In der `REDIRECT`-Nachricht kann zusätzlich das nächste Gateway angegeben werden (Feld „Zwischenziel“ in Abbildung 5.17), sodass das vorherige Gateway nicht versucht, einen neuen Pfad zur Senke, sondern nur einen Pfad zum nächsten Gateway zu schalten. Der Abschnitt des alten Pfades vom nächsten Gateway bis zur Senke bleibt dadurch erhalten, und nur das aktuelle Gateway wird durch einen neuen Abschnitt ersetzt. Falls der nächste Pfadabschnitt nicht verwendet werden kann, weil kein Pfad gefunden wird, so muss er freigegeben werden. Das geschieht entweder über ein Timeout oder über eine Nachricht vom vorherigen Gateway an das nächste. Das Timeout kann vor dem Versenden der `REDIRECT`-Nachricht durch eine `KEEPALIVE`-Nachricht, passend für die Suche der Umleitung, eingestellt werden.

Die `REDIRECT`-Nachricht kann auch bei der Pfadsuche und der Pfadschaltung für das Vorschlagen eines alternativen Gateway verwendet werden.

Der teilweise oder vollständige Abbau der Verbindung wird über die `DISCONNECT`-Nachricht gesteuert (Abbildung 5.18). Der vollständige Abbau der Verbindung geht immer vom Initiator-Gateway aus. Zu einem teilweisen Abbau der Verbindung kann es, wie oben beschrieben, kommen, wenn ein Gateway ausfällt und die Gateways auf dem vom Initiator-Gateway abgeschnittenen Teil des Pfades keine Nachrichten mehr vom Initiator-Gateway erhalten.

Die Kommunikationsmuster von `DISCONNECT` und die assoziierten Vorgänge auf den Gateways sind in Abbildung 5.19 zu sehen:

1. Der Abbau der Verbindung wird entweder durch eine eintreffende Nachricht (`DISCONNECT`) oder durch einen Fehler, der den Verbindungsabbau notwendig macht, ausgelöst.

2. Der Verbindungsabbau wird quittiert und weitergeleitet (`DISCONNECT` und `DISCONNECTACK`) und die Ressourcen und Adressen werden freigegeben sowie das Rewriting deaktiviert; bis zur Bestätigung des Verbindungsabbaus in Schritt 3 und bis zur internen Bestätigung, dass die Ressourcen freigegeben wurden und das Address-Rewriting deaktiviert wurde, bleibt die Verbindung im Zustand `CLOSING`.

3. Der Verbindungsabbau wird quittiert (`DISCONNECTACK`).

Nach Schritt 3 und der Bestätigung der Freigabe der Ressourcen und der Adressen kann die Verbindung gelöscht werden.

ERROR
Initiator-Gateway
Verbindungs-ID
Empfängeradresse
Absenderadresse
Zeitstempel des Absenders
Fehler-Code: zu definieren für verschiedene Timeouts, Ausfälle und Kommunikationsfehler (weitere Fehler-Codes können zusammen mit QoS-Parametern definiert werden)
Fehlermeldung (menschenlesbarer Text)
Signatur des Absenders (optional)

REDIRECT
Initiator-Gateway
Verbindungs-ID
Empfängeradresse
Absenderadresse
Zeitstempel des Absenders
Ersatz-Gateway (optional)
Gateway-Zwischenziel (optional)
Signatur des Absenders (optional)

ERRORACK / REDIRECTACK
Initiator-Gateway
Verbindungs-ID
Empfängeradresse
Absenderadresse
Zeitstempel der bestätigten Nachricht
Signatur des Absenders (optional)

Abbildung 5.17.: Die PPCP-Nachrichten ERROR, REDIRECT, ERRORACK und REDIRECTACK

DISCONNECT
Initiator-Gateway
Verbindungs-ID
Empfängeradresse
Absenderadresse
Zeitstempel des Absenders
Signatur des Absenders (optional)

DISCONNECTACK
Initiator-Gateway
Verbindungs-ID
Empfängeradresse
Absenderadresse
Zeitstempel der bestätigten Nachricht
Signatur des Absenders (optional)

Abbildung 5.18.: Die PPCP-Nachrichten DISCONNECT und DISCONNECTACK

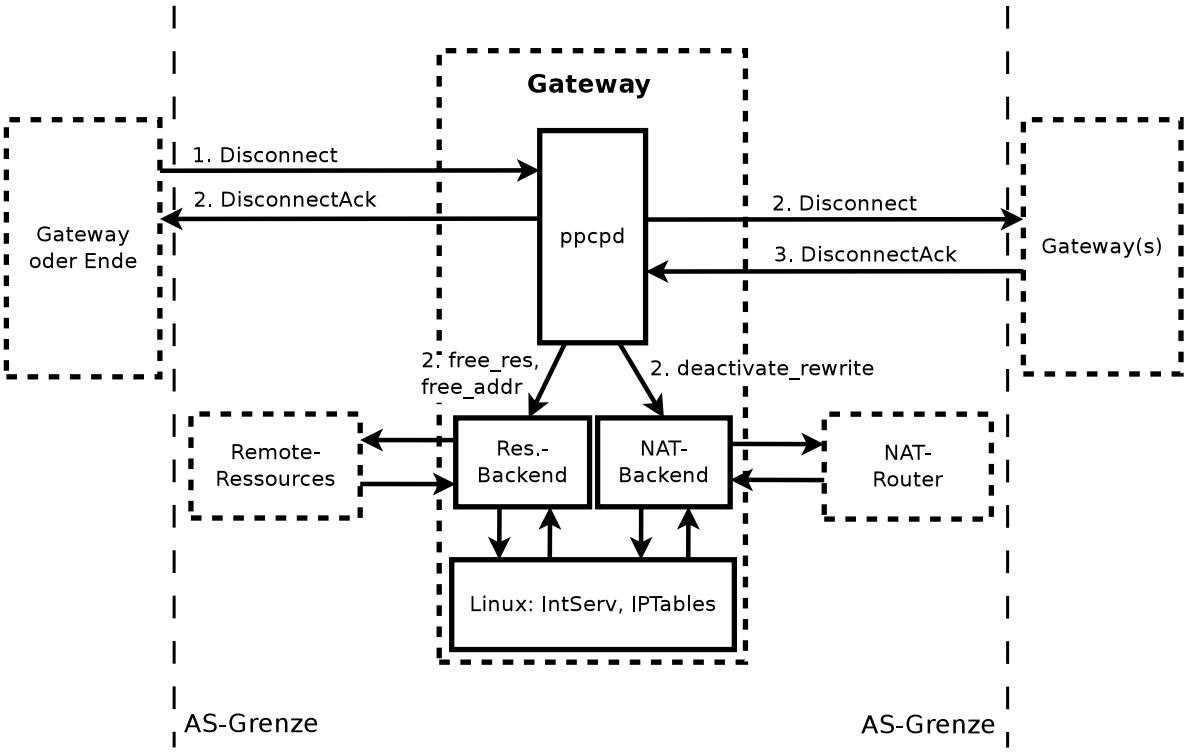


Abbildung 5.19.: Kommunikationsmuster von `DISCONNECT`

5.3. Erweiterungen von PPCP

5.3.1. PPCP-Nachrichten: `INFO`, `UPDATE` und `CONFIRM`

Das Path-Parameter-Control-Protocol kann die Eigenschaften (insbesondere die QoS-Parameter) bestehender Verbindungen nicht ändern. Wie in den Szenarien in Kapitel 2 gesehen wurde, wäre diese Möglichkeit aber sehr nützlich. Die Spezifizierung einer `UPDATE`-Nachricht wäre nicht sehr aufwändig, auf den ersten Blick wären die Felder von SEARCH dafür geeignet. Schwieriger ist allerdings die Konzeption der Umsetzung: Wenn einzelne Gateways die neuen Anforderungen nicht erfüllen können, müssen neue Pfade und Umleitungen gesucht werden. Dafür stehen mit `REDIRECT` zwar schon Mittel zur Verfügung, allerdings ist fraglich, ob es nicht einfacher ist, ausgehend vom Initiator-Gateway direkt einen neuen Pfad zu suchen. Davon wäre nur die Schnittstelle zu den Verbindungsnutzern betroffen, und das Initiator-Gateway könnte den Wechsel zur neuen Verbindung für den Verbindungsnutzer transparent vollziehen. Ob die Erweiterung von PPCP um eine `UPDATE`-Nachricht sinnvoll ist, müsste sich im Einsatz zeigen.

Eine Lücke, die nicht direkt die Pfadschaltung betrifft, würde eine `INFO`-Nachricht füllen, anhand derer Informationen über Gateways ausgetauscht werden könnten. Mechanismen zur Bekanntmachung neuer Gateways, zur Änderung von Gateway-Adressen oder zur Bekanntmachung von QoS-Netzen oder unterstützten QoS-Parametern wären eine sinnvolle Erweiterung für das PPCP, könnten aber auch mit einem eigenen Protokoll gelöst werden.

In diesem Kapitel wird an verschiedenen Stellen (unter anderem in Abschnitt 5.3.3 vorgeschlagen, eine PPCP-Nachricht zur Bestätigung an das Initiator-Gateway zu senden, ohne dafür einen Mechanismus anzugeben. Dazu könnte bei der Ausarbeitung der Pfadschaltungssicherheit eine `CONFIRM`-Nachricht spezifiziert werden, die als Container der zu überprüfenden PPCP-Nachricht dient.

5.3.2. Multicast und Multidirektionalität

Multicast und Multidirektionalität wurden in Kapitel 2 in Szenario 2.1.3 angesprochen. Eine Verallgemeinerung der dort vorgestellten Multicast-Verbindungen lässt zu, dass bei bestehender Verbindung weitere Senken hinzugefügt und entfernt werden können und entsprechend Teilpfade gesucht, geschaltet und wieder entfernt werden. Eine Multicast-Verbindung besteht dann aus einer oder mehreren Quellen und keiner, einer oder mehreren Senken. Es werden auch gar keine Senken zugelassen, damit eine Multicast-Verbindung nicht automatisch mit der letzten Senke, sondern nur explizit gelöscht wird. Eine Senke als (potentieller) Verbindungsnutzer soll aber Voraussetzung für die Erhaltung einer Verbindung sein.

Für die Pfadsuche müsste man die `SEARCH`-Nachricht um weitere Felder für Quellen und Senken erweitern und Multicast als Verbindungsoption einführen. Die Gateways müssten dann entscheiden, welche Quellen und welche Senken an welches Gateway zur Suche weitergeleitet würden. Sollen Verbindungsendpunkte hinzugefügt oder entfernt werden, so können dazu die aktuellen Nachrichten verwendet werden. Die Gateways erkennen an den in der Nachricht angegebenen Quellen und Senken, ob die ganze Verbindung erweitert oder abgebaut werden soll, oder nur ein Teil davon.

Werden Multicast und Bidirektionalität als Verbindungsoption gewählt, so wird, wie bisher bei den bidirektionalen Verbindungen, je eine Rewriting-Adresse für die Richtung Quelle-Senke und eine für die Richtung Senke-Quelle reserviert und auf dem letzten Gateway jeweils die Absenderadresse gegen eine Rewriting-Adresse ausgetauscht. Der Absender ist in diesem Fall eine Gruppe (die Multicast-Gruppe) von Quellen, die Semantik von „Antworten“ hat sich dadurch geändert: Die „Frage“ stellt ein Einzelner, die „Antwort“ erhalten alle. Das ist, wie das Szenario „Videokonferenz“ zeigt, eine sinnvolle Eigenschaft für eine PPCP-Verbindung. Diese Art von Multicast-Verbindung wurde in Kapitel 2 „multidirektional“ genannt.

Das PPCP ist also grungsätzlich so angelegt, dass es um eine Multicast-Option erweitert werden kann. Die dazu notwendige Anpassung des Address-Rewriting ist zwar schon möglich, die geeigneten Techniken (z. B. TEE-Regeln bei iptables) sind aber (noch) in der Entwicklung und nicht etabliert, sodass ihre Integration im Rahmen dieser Arbeit nicht möglich war. So fließen hier nur einige Gedanken dazu ein, die sich durch die Beschäftigung mit dem Thema ergaben.

5.3.3. Sicherheitsaspekte von PPCP

Wie im Rahmen der Skizzierung einiger Sicherheitsaspekte der Pfadschaltung in Abschnitt 4.7 angedeutet, liegen die Ausarbeitung und Umsetzung von Sicherheitskonzepten für die Pfadschaltung außerhalb dieser Arbeit. Zumindest die Authentifizierung der Absender und des Verbindungsbesitzers sollen aber in Form von Signaturfeldern in den Nachrichten vorgesehen werden, sodass die Gateway-Betreiber die Möglichkeit haben, eventuell schon vorhandene Infrastruktur (z. B. CA-signierte Schlüssel) direkt zu nutzen. Zeitstempel sind ohnehin Teil des Protokolls, und das Senden von Pfadschaltungsnachrichten an das Initiator-Gateway zur Bestätigung ist ebenfalls angedeutet (Abschnitt 5.3.1). Hier werden noch einige Punkte angesprochen, die bei der Definition der Nachrichten übergangen wurden und Hinweise auf eine mögliche Ausarbeitung der Sicherheit von PPCP geben sollen.

Die Signatur des Initiator-Gateway dient dazu, den Verbindungsbesitzer bei der Pfadsuche oder Pfadschaltung (beides geht ursprünglich vom Initiator-Gateway aus) zu authentifizieren. Alle Gateways, die an einer Pfadschaltung beteiligt sind, haben den Verbindungsbesitzer zusammen mit ihren Nachbarn authentifiziert. Sie haben eine Nachricht erhalten, die eine Signatur des Initiator-Gateways enthält, und diese Signatur hat das Nachbar-Gateway signiert. Das Nachbar-Gateway hat damit bestätigt, dass es den Verbindungsbesitzer kennt bzw. anhand der Signatur authentifizieren kann. Die Nachbarn wissen in diesem Fall also, wessen Verbindung sie steuern.

Nicht jede Nachricht geht ursprünglich vom Initiator-Gateway aus, daher kann auch nicht jede vom Initiator-Gateway signiert werden. Manche Entscheidungen müssen Gateways lokal fällen. Sie sind damit auch für die Konsequenzen verantwortlich. Sie handeln nicht, wie bei der Pfadsuche, im „signierten“ Auftrag des Initiator-Gateway. `ERROR`- und `REDIRECT`-Nachrichten sind Beispiele für Nachrichten,

die die Verbindung verändern und unterbrechen können, ohne dass das Initiator-Gateway involviert ist.

Die Signatur des Initiator-Gateways dient also nicht der Autorisierung aller PPCP-Nachrichten einer Verbindung, sondern der Identifikation des Besitzers. Die Kontrolle der Verbindung ist auf alle Gateways einer Verbindung verteilt, eine Gateway-Chain-of-Trust ist daher Voraussetzung für den Einsatz. Die Chain-of-Trust besteht hier aus den Gateways, die wissen, wessen Verbindung sie steuern (und wer möglicherweise für die Kosten aufkommt).

Der signierte Zeitstempel des Initiator-Gateways bzw. die Begrenzung der Gültigkeitsdauer von Nachrichten ist nicht immer sinnvoll. Muss ein Gateway einen neuen Teilpfad suchen, so muss es entweder vorher eine signierte `SEARCH`-Nachricht gespeichert haben, die dann einen alten Zeitstempel trägt, oder die `SEARCH`-Nachricht nur selbst signieren, sodass das Initiator-Gateway nicht mehr authentifiziert werden kann. Letzteres bedeutet, dass das Gateway, das die `SEARCH`-Nachricht empfängt, dem Absender auf einem dritten Level vertrauen muss, es ist (noch) nicht Teil der Chain-of-Trust.

Die drei Vertrauenslevel für Nachrichten sind:

1. Nachrichtenteile wurden vom Initiator-Gateway signiert, und die ganze Nachricht wurde vom Absender signiert.
2. Die Nachricht wurde von einem Absender signiert, der das Initiator-Gateway authentifiziert hat und weiß, dass der Empfänger das Initiator-Gateway ebenfalls authentifiziert hat (Chain-of-Trust, Nachrichtenbeispiel: `ESTABLISHED`).
3. Die Nachricht wurde vom Absender signiert, und der Empfänger hat noch keine gesicherten Informationen über den Initiator.

Eine Möglichkeit, um neue Gateways in die Chain-of-Trust aufzunehmen, ist die Bestätigung abgelaufener oder nur vom Nachbar-Gateway signierter Nachrichten durch das Initiator-Gateway. Eine geeignete `CONFIRM`-Nachricht wurde im Abschnitt 5.3.1 vorgeschlagen. Hier sei noch einmal der Hinweis wiederholt, dass dieser Abschnitt *keine Lösung der PPCP-Sicherheit* ist, sondern nur die Beschreibung einer möglichen Authentifizierungsstruktur, die den Kommunikationsabläufen des PPCP inhärent ist.

Damit ist die Spezifikation des Path-Parameter-Control-Protocol und die Skizzierung einiger Erweiterungsmöglichkeiten abgeschlossen. Im folgenden Kapitel wird die prototypische Implementierung vorgestellt.

6. Prototypische Implementierung und Auswertung

In diesem Kapitel wird ein Überblick über die prototypische Implementierung des PPCP-Dienstes (`ppcpd`) für die Pfadschaltung gegeben und die virtuelle Testumgebung für die Auswertung vorgestellt.

Die Implementierung wurde in der Programmiersprache C geschrieben und auf der Linux-Distrubution Debian mit dem gcc-Compiler und flex entwickelt. Für das Address-Rewriting wurde textttiptables verwendet. Für die Testumgebung kamen unter anderem User-Mode-Linux und Xen auf Debian sowie die Bridge-Utils für die Simulation der Links zum Einsatz.

6.1. Der Path-Parameter-Control-Protocol-Daemon (`ppcpd`)

Abbildung 6.1 zeigt die Module von `ppcpd` und deren wichtigsten Funktionen:

ppcpd: `ppcpd.c` enthält die main-Funktion, führt die Initialisierung aus und wartet in einer Schleife auf UDP-Pakete mit PPCP-Nachrichten

protocol: Das protocol-Modul enthält einen für flex geschriebenen Parser (`protocol_parser.l`) für PPCP-Nachrichten, und Funktionen für das Auswerten und Senden von PPCP-Nachrichten (`protocol.c`)

condb: Das condb-Modul dient der Speicherung von Verbindungsinformationen, d. h. dem Verbindungszustand, der ID, dem Besitzer, dem vorherigen und dem (bzw. den) nächsten Gateway(s) zugeordneten Ressourcen und Adressen etc. (`condb.c`)

gwhandling: Das gwhandling-Modul enthält die Funktion für das Gateway-Lookup (`gwhandling.c`) und für die Aktualisierung der Gateway-Informationen (`gwfile_parser.l`)

qos: Das qos-Modul enthält einen Parser für QoS-Parameter (`qos_parser.l`) und Funktionen, um aus den QoS-Parsern eine Metrik zu berechnen (`qos.c`)

res: Das res-Modul enthält Funktionen zur Anforderung und Freigabe von Ressourcen und Adressen sowie Funktionen zur (entfernten oder lokalen) Aktivierung und Deaktivierung des Address-Rewriting (`res.c`)

net: Das net-Modul enthält Funktionen zur Erstellung verschiedener Sockets und zum Senden von Strings bzw. PPCP-Nachrichten (`net.c`)

auth: Das auth-Modul enthält Funktionen zur Authentifizierung von Absender und Initiator der PPCP-Nachrichten (`auth.c`)

config: Das config-Modul enthält einen Parser für die Konfigurationsdateien (`config.l`)

logging: Das logging-Modul enthält Funktionen für das Logging und Debugging (`logging.c`)

util: Das util-Modul enthält allgemeine Hilfsfunktionen (`util.c`)

Nicht alle hier beschriebenen Funktionen wurden auch implementiert. Die Ressourcenverwaltung und die Berechnung der Metrik werden nur simuliert, qos.c und res.c sind nur eine Schnittstelle zu den QoS-Backends, welche für jede QoS-Technologie implementiert werden müssen. Für die Authentifizierung gilt das Gleiche, ein spezifischer Authentifizierungsmechanismus ist nicht definiert und muss noch implementiert werden. `config`, `logging` und `util` werden in fast allen Modulen verwendet, daher wurden auf der Abbildung die Abhängigkeiten nicht eingezeichnet.

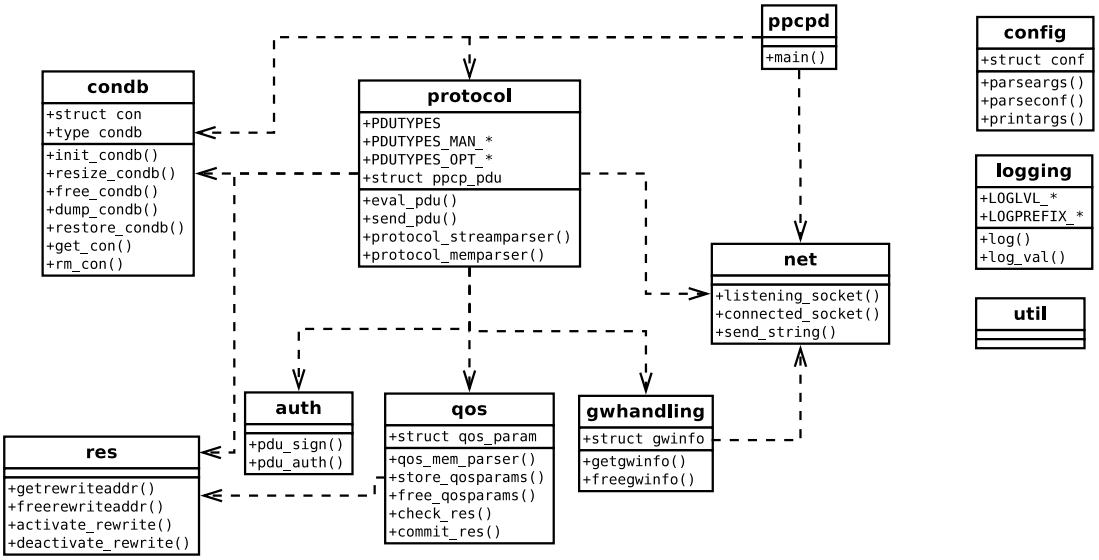


Abbildung 6.1.: Die Module der ppcpd-Implementierung

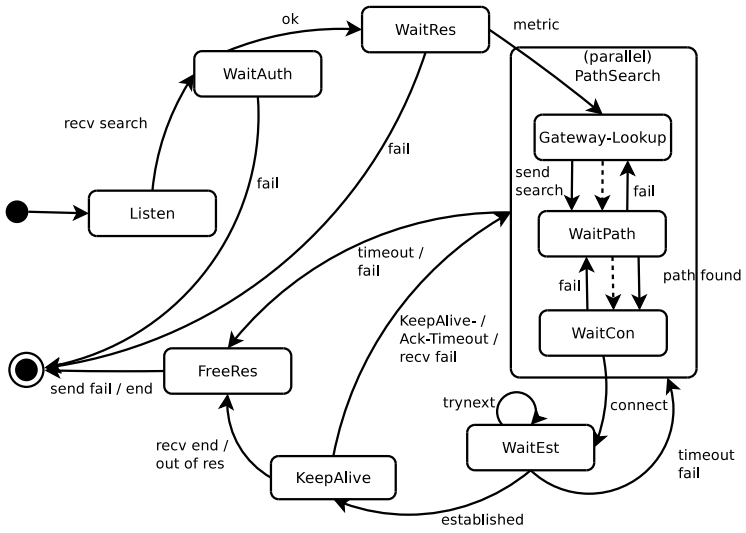


Abbildung 6.2.: Zustandsdiagramm einer ppcpd-Instanz

In Abbildung 6.2 ist ein Zustandsdiagramm für eine ppcpd-Instanz bezüglich einer Verbindung dargestellt. Einige der Zustände sind mit den entsprechenden Aktionen der Module verknüpft. Im Zustand WaitAuth wird auf die Authentifizierung der SEARCH-Nachricht gewartet. Zugunsten der Übersichtlichkeit wurde auf einen WaitAuth-Zustand für die anderen Nachrichten verzichtet Das Eintreffen jeder Nachricht verursacht einen Übergang zu WaitAuth. Ist die Authentifizierung nicht erfolgreich, so wird die Nachricht ignoriert und der vorherige Zustand wiederhergestellt, andernfalls geht es weiter, wie auf der Abbildung zu sehen ist.

Die wichtigsten Module werden in den folgenden Abschnitten genauer erläutert.

6.1.1. Das protocol-Modul

Die per UDP eintreffenden Nachrichten werden in `main` an den Protokoll-Parser übergeben. Die Funktion `protocol_memparser` parst im Speicher abgelegte Nachrichten, während für Datenströme (Dateien, TCP-Sockets oder Pipes) die Funktion `protocol_streamparser` zur Verfügung steht.

Wenn ein Authentifizierungsmechanismus implementiert ist, werden die Nachrichten vor dem Parsen authentifiziert. Die Nachrichten folgen, bis auf die Kommentare, der Syntax von JSON[1]. Beispiele für die PPCP-Nachrichten sind in Anhang B zu finden; Felder, die in der Implementierung nicht verwendet werden, sind auskommentiert. In einigen Nachrichten im Anhang sind Erweiterungen für Multicast angegeben, diese Angaben sind nicht vollständig.

Die Nachrichten werden geparst und in einer für alle Nachrichten einheitlichen Struktur (`struct ppcp_pdu`) gespeichert. Felder, die in der Nachricht nicht vorkommen, werden als leer markiert (mit 0 oder `NULL`). Die Struktur wird ausgewertet (`eval_pdu`) und einer Verbindung zugeordnet (siehe Abschnitt 6.1.3). Je nach Zustand der Verbindung erfolgt eine andere Reaktion. Treffen fehlerhafte oder im aktuellen Zustand unerwartete Nachrichten ein, so wird mit einer Fehlernachricht geantwortet, andernfalls wird die passende Aktion ausgelöst und eine Antwortnachricht erstellt und versendet.

6.1.2. Das gwhandling-Modul (Gateway-Lookup)

Die zentrale Funktion dieses Moduls, die den Pfadverlauf wesentlich beeinflusst, ist `getgwinfo`, die den Gateway-Lookup implementiert. Aus einer Zieladresse wird mit Hilfe einer Datenbank eine verkettete Liste von Strukturen erstellt, die Informationen zu den Gateways enthalten. Die Gateways kommen als nächstes Gateway auf dem Pfad in Frage und sind absteigend nach Priorität sortiert. Für das erste Gateway in der Liste wird ein UDP-Socket erstellt, der sofort zum Senden von Nachrichten verwendet werden kann.

`getgwinfo` muss auch erkennen, ob die Zieladresse der Pfadsuche im Zuständigkeitsbereich des aktuellen Gateway liegt.

Wird kein passendes Gateway gefunden, so wird die Datenbank mit den Gateways aktualisiert. In der hier beschriebenen Implementierung ist das durch das Parsen einer Tabelle (siehe Anhang C) umgesetzt, an dieser Stelle kann aber auch eine Schnittstelle zu BGP-Speakern hinzugefügt werden. Die Aktualisierung der Datenbank kann auch manuell durch das Signal `SIGHUP` ausgelöst werden.

Die Liste mit den Gateways wird zusammen mit dem Verbindungszustand gespeichert, da die Gateways während der Lebensdauer der Verbindung verwendet werden können.

Mit dem Löschen der Verbindung werden die offenen Sockets geschlossen und die Liste gelöscht (`freegwinfo`).

[1] http://www.json.org

6.1.3. Das condb-Modul

Das Modul `condb` bietet Funktionen für das Finden, Anlegen und Löschen von Verbindungsdatensätzen (`get_con` und `rm_con`) sowie für die Datenbankverwaltung (`init_condb`, `resize_condb`, `free_condb`, `dump_condb` und `restore_condb`).

Zur Generierung und Verarbeitung der Verbindungs-IDs wird `libuuid` aus dem util-linux-ng-Paket verwendet. Die ID (UUID bzw. Universally-Unique-Identifier) besteht aus 128 Bit, und man kann „vernünftigerweise davon ausgehen“[2], dass sie global eindeutig ist. Für garantierte Eindeutigkeit der Verbindungsidentifizierung kann als Schlüssel das Paar aus Initiator-Gateway und UUID verwendet werden. Auf dem Initiator-Gateway muss dann zusätzlich über die verwendeten UUIDs Buch geführt werden.

`get_con` sucht anhand einer Verbindungs-ID die Struktur, die die Verbindungsdaten enthält. Existiert keine solche Struktur, so wird sie von `get_con` angelegt. Die Verbindungsdaten werden direkt in dieser Struktur geändert. `rm_con` löscht Verbindungen und „räumt“ die enthaltenen Datenstrukturen „auf“ (schließt Sockets, gibt Speicher frei etc.).

Die Funktionen `dump_condb` und `restore_condb` dienen dem Backup und der Wiederherstellung von Verbindungsdaten, um, wie in Abschnitt 4.6 beschrieben, nach einem Ausfall die Informationen über belegte Ressourcen und aktive Rewritings wiederherstellen zu können.

`init_condb` legt eine Datenbank an und initialisiert sie, `resize_condb` vergrößert sie und `free_condb` löscht sie bzw. gibt den Speicher frei.

6.1.4. Das qos-Modul

Das qos-Modul bietet (implementierte) Funktionen zum Parsen der QoS-Parameter aus PPCP-Nachrichten (`qos_mem_parser`), sowie zum Speichern der geparsten Parameter (`store_qosparams`) und zum Löschen bzw. zur Freigabe des durch die QoS-Parameter belegten Speichers (`free_qosparams`).

Der Parser unterstützt die in Abschnitt 4.4.2 definierten QoS-Parameter. Die Syntax ist in den PPCP-Nachrichten in Anhang B zu sehen.

Nicht durch PPCP unterstützte QoS-Parameter können in Form von JSON-Objekten in die PPCP-Nachrichten eingebettet und vom Parser gespeichert werden. Nach dem Parsen werden die QoS-Parameter an die Funktion `check_res` übergeben, um daraus, im Zusammenspiel mit den QoS-Backends (über das `res`-Modul), eine Metrik zu berechnen und Ressourcen vorzumerken oder zu reservieren. Sobald die Verbindung steht, werden die Ressourcen mit `commit_res` einer Verbindung zugeordnet.

Dieser Teil der Implementierung ist nicht ausgearbeitet, die Funktionen `check_res` und `commit_res` müssen zusammen mit den QoS-Backends implementiert werden. Für das Testen der Pfadsuche gibt `check_res` eine zufällige Metrik zurück, die auf die in Abschnitt 6.2 beschriebene Testumgebung ausgelegt ist.

6.1.5. Das res-Modul und der Rewrite-Daemon (`rewrited`)

Das res-Modul implementiert die Schnittstelle zu den Ressourcen. Es vergibt Rewriting-Adressen aus dem in der Konfigurationsdatei angegebenen QoS-Netz (`getrewriteaddr` und `freerewriteaddr`) und setzt das Address-Rewriting um (`activate_rewrite` und `deactivate_rewrite`).

Wie in Abschnitt 4.5 ausgeführt ist, verwaltet je ein Gateway seinen eigenen Address-Pool, die Vergabe und Freigabe von Rewriting-Adressen wird im res-Modul entsprechend lokal umgesetzt. Der ebenfalls dort beschriebene Rewriting-Router muss aber nicht das Gateway sein, daher ist die Aktivierung und Deaktivierung des Address-Rewriting in einen eigenen Dienst ausgelagert, den `rewrited`. Die Adresse, über die der `rewrited` erreicht werden kann, ist in der Konfigurationsdatei einstellbar.

[2] man uuid

Werden mehrere QoS-Netze verwendet, so ist es sinnvoll, jedem QoS-Netz einen Rewriting-Router zuordnen zu können. Das bringt aber auch eine höhere Komplexität des Gateway-Lookup mit sich, da die Rewriting-Router und die von ihnen gerouteten QoS-Netze nach Abschnitt 4.5 topologisch bedingt unterschiedliche Nachbar-Gateways bedienen. In diesem Fall würden mit dem Gateway-Lookup auch der passende Rewriting-Router und das zugehörige QoS-Netz ermittelt. Die prototypische Implementierung ist darauf angelegt, dass jedes Gateway nur einen Rewriting-Router verwaltet.

Das Address-Rewriting des `rewrited` wird durch `iptables`[3] umgesetzt. Der Rewriting-Daemon führt für die Aktivierung und Deaktivierung folgende Befehle aus:

```
1 iptables -t nat -A PREROUTING -d $local-rewriting-address -j DNAT \
2          --to-destination $next-rewriting-address
3 iptables -t nat -D PREROUTING -d $local-rewriting-address -j DNAT \
4          --to-destination $next-rewriting-address
```

Die Reservierung und Zuordnung von Ressourcen, auf die das qos-Modul und das protocol-Modul zurückgreifen, ist nicht ausgearbeitet.

6.2. Test und Auswertung

Um das Path-Parameter-Control-Protocol und den `ppcpd` zu testen, werden autonome Systeme, Gateways, Border- und Rewrite-Router sowie Verbindungsenden benötigt. Abschnitt 6.2.1 beschreibt Entwurf und Umsetzung der Testumgebung.

Die Konfiguration und der Test des `ppcpd` werden in Abschnitt 6.2.2 beschrieben.

6.2.1. Aufbau der Testumgebung

Die Testumgebung besteht aus sieben autonomen Systemen, die alle mindestens ein Gateway bereitstellen, einige autonome Systeme enthalten auch normale Router sowie Endpunkte. Abbildung 6.3 gibt einen Überblick über die Topologie.

Um nicht von einer Virtualisierungsplattform abhängig zu sein, bestehen die autonomen Systeme auf Rechner 1 (AS0 - AS4) aus User-Mode-Linux-Instanzen und auf Rechner 2 aus Xen-VMs. Die VMs und Systeme werden über virtuelle Ethernet-Bridges verbunden, wobei die Ethernet-Verbindung zwischen den beiden Hardware-Rechnern ebenfalls auf beiden Rechnern an eine Bridge angeschlossen wird.

Eine Bridge repräsentiert je einen Link (bzw. eine Menge verbundener Links) auf der zweiten Schicht des OSI-Modells. Die Bridges sind in Abbildung 6.3 eingezeichnet, das Skript zur Erstellung und Konfiguration ist in Anhang C aufgeführt.

Über ein Subversion-Repository wurden Konfigurationsdateien und Setup-Skripte für alle VMs verwaltet. Sie sind teilweise in Anhang C aufgeführt.

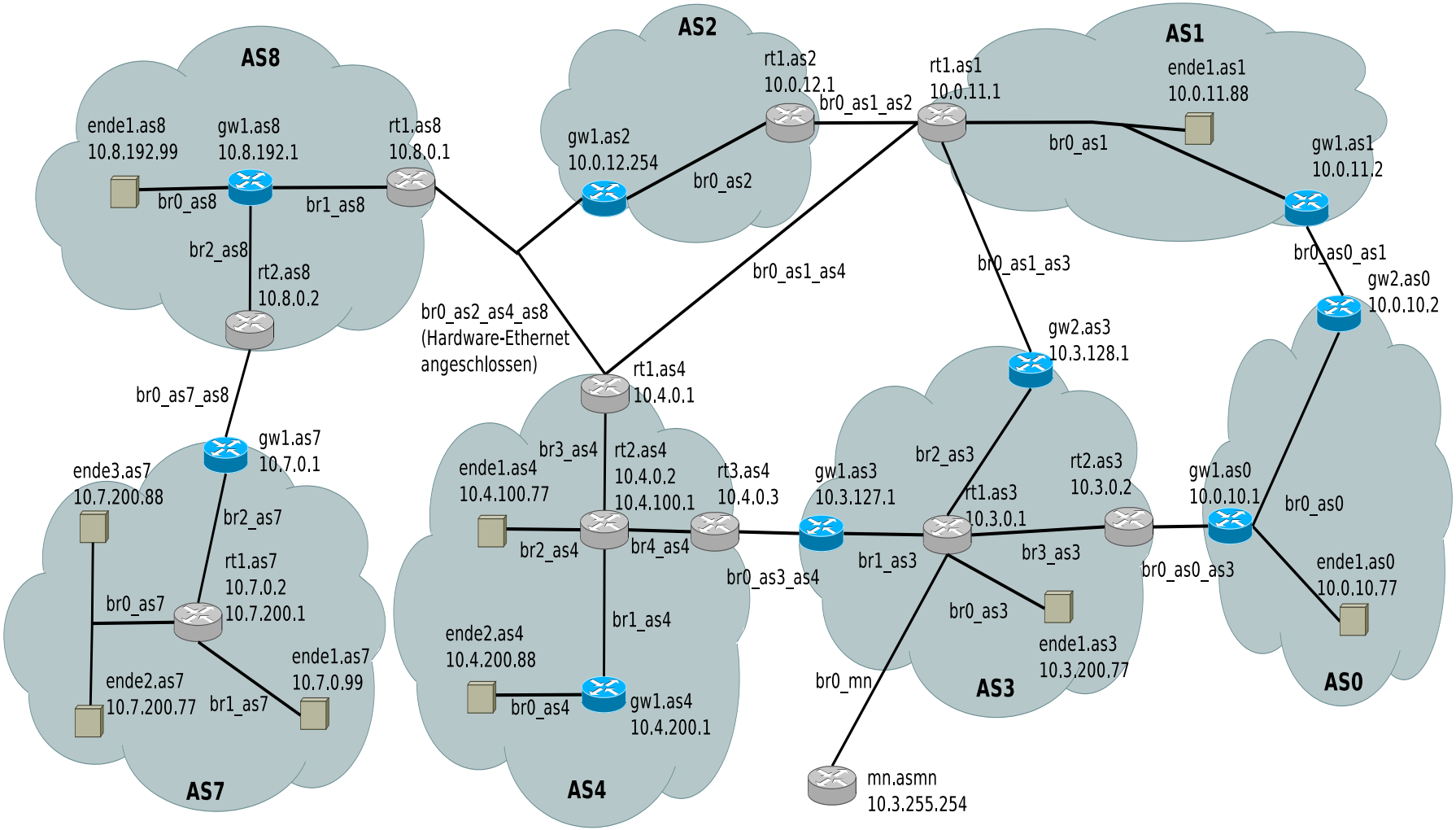
6.2.2. Test und Auswertung

Auf jedem Gateway der Testumgebung wurde eine Konfigurationsdatei für den `ppcpd` und eine Tabelle für den Gateway-Lookup angelegt.

Allen Gateways ist ein eigenes QoS-Subnetz zugeordnet, und sie verwenden einen lokal laufenden `rewrited` für das Address-Rewriting. Konfigurationsbeispiele sind ebenfalls in Anhang C zu sehen.

[3] http://www.netfilter.org/

Abbildung 6.3.: Versuchsnetz-Topologie, AS7 und AS8 bestehen aus Xen-VMs auf Rechner 1, die anderen autonomen Systeme bestehen aus User-Mode-Linux-Instanzen auf Rechner 2; die Router mit „gw“ im Namen haben Gateway-Funktion, mn.asmn ist eine Management-VM und die Verbindung der VMs zur Aussenwelt



Auf allen Gateways wird der `ppcpd` und der `rewrited` installiert und gestartet, anschließend wird auf einem Endpunkt eine `CONNECT`-Nachricht an das zuständige Gateway gesendet. Als Client kommt netcat6[4] (nc) zum Einsatz:

```
1 cat connect.ppcp | nc -4 -u gw1.as0 7429 -q 1
```

Ein Beispiel für eine `CONNECT`-Nachricht ist in Anhang B zu sehen, ebenso für die Antwortnachrichten. Die Antworten des Gateway können ebenfalls mit netcat (in einer Schleife) gelesen werden.

War die Pfadschaltung erfolgreich, so trifft mit der `ESTABLISHED`-Nachricht die Verbindungsadresse ein. Die zufällige Metrik, die in jedem Gateway berechnet wird, ist so dimensioniert, dass die Pfadschaltung meistens erfolgreich ist.

Die Verbindungsadresse wurde mit `ping` und `ssh` getestet. Bei einer unidirektionalen Verbindung erhält man ping-Antworten mit einer anderen Absenderadresse (der eigentlichen Adresse des Absenders, nicht der Verbindungsadresse, an die die Ping-Nachrichten gesendet wurden). Die Rewriting-Aktionen der einzelnen Gateways können mit `iptables -tnat -L` verfolgt werden. Der Test des Prototyps verlief erfolgreich, die Pfade wurden geschaltet und konnten von Anwendungen genutzt werden.

[4] http://www.deepspace6.net/projects/netcat6.html

7. Fazit und Ausblick

Eine Ende-zu-Ende-Verbindung über das Internet mit einer spezifischen Dienstgüte erfordert die Schaltung eines Pfades, um Ressourcen entlang dieses Pfades für die Gewährleistung der Dienstgüte zu reservieren. Die dienstgütespezifische Pfadschaltung in kleineren, einheitlich verwalteten IP-Netzen ist mit vorhandenen Technologien möglich. Im Internet scheitert sie einerseits an der Heterogenität der Verwaltung und der eingesetzten Technologie und andererseits an den begrenzten Möglichkeiten des IP-Routing.

Welche Technologien in einem autonomen System zum Einsatz kommen und wem ihre Fähigkeiten zur Verfügung stehen, liegt in der Hand der Betreiber. Für die Pfadschaltung bedeutet das, dass die vorhandenen QoS-Technologien aller autonomen Systeme, die von Ende zu Ende passiert werden, für den Pfad gekoppelt werden müssen und dass die Betreiber dieser autonomen Systeme dem Verbindungsnutzer auch Zugriff auf die Technologien gewähren müssen.

Hinzu kommt, dass mit normalem IP-Routing die Schaltung von Ende-zu-Ende-Pfaden nicht realisierbar ist. Dazu wäre es nötig, für jeden Pfad auf jedem beteiligten Router einen Eintrag in die Routing-Tabelle vorzunehmen oder zu überprüfen, ob der aktuelle Eintrag für den Pfad korrekt ist und sich während der Verbindung nicht ändern kann. In der Realität ist es oft noch nicht einmal möglich, die Router zu bestimmen, die an der Kommunikation von Ende zu Ende beteiligt sind. Zugriffe auf fremde Routing-Tabellen sind erst recht ausgeschlossen.

Diese Arbeit gibt eine Lösung für das Pfadschaltungsproblem an, die ohne eine Anpassung des Routing auskommt und keine fremden Eingriffe in die Infrastruktur der autonomen Systeme erfordert. Der Zugriff auf die Ressourcen wird durch ein eigenes Pfadschaltungs-Gateway in jedem autonomen System gesteuert. Die Schaltung der Pfade beruht darauf, dass für das Routing einer Verbindung nicht die Adresse eines der Verbindungsendpunkte, sondern auf jedem Abschnitt des Pfades eine andere Empfängeradresse verwendet wird. Nur auf dem letzten Abschnitt wird die Empfängeradresse des Verbindungsendpunktes verwendet. Die Empfängeradresse muss in jedem autonomen System an einem Punkt ausgetauscht werden (Address-Rewriting). Auf allen Routern zwischen diesen Punkten werden nur „normale“ Einträge in die Routing-Tabellen benötigt, die unabhängig von einzelnen Verbindungen bestehen. Die Einträge in den Routing-Tabellen betreffen nur benachbarte autonome Systeme, für entfernte Netze werden keine zusätzlichen Routing-Informationen benötigt. Ebenso wird bei der Pfadsuche keine Kenntnis der gesamten Topologie vorausgesetzt, sondern nur über die Kenntnis der benachbarten Gateways.

In Kapitel 4 werden die Konzepte für die Pfadschaltung durch die Gateways ausgearbeitet und in Kapitel 5, darauf aufbauend, das Path-Parameter-Control-Protocol zur Suche und Schaltung von dienstgütespezifischen Pfaden durch die Gateways für Ende-zu-Ende-Verbindungen spezifiziert. Für das Protokoll wird in Kapitel 6 eine prototypische Implementierung und eine Testumgebung vorgestellt.

Voraussetzung für den Einsatz des PPCP sind vorhandene QoS-Mechanismen innerhalb der autonomen Systeme. Der Zugriff auf die Ressourcen durch die Gateways geschieht über QoS-Backends, die auf die QoS-Mechanismen der autonomen Systeme aufsetzen und die Verfügbarkeit von Ressourcen sowie die Anforderungen einer Verbindung bewerten und bei einer erfolgreichen Pfadschaltung die Ressourcen im eigenen autonomen System der Verbindung zuordnen. Auf diese Weise wird von der eingesetzten QoS-Technologie abstrahiert, und der Zugriff auf die eigenen Ressourcen bleibt in der Hand der Betreiber. Durch die Unabhängigkeit von der eingesetzten QoS-Technologie kann mit der Implementierung eines QoS-Backends prinzipiell jedes autonome System an der Pfadschaltung teilnehmen. Auch neue Entwicklungen können in die Pfadschaltung integriert werden.

Das Path-Parameter-Control-Protocol ermöglicht bei der Pfadsuche den Austausch von durch das Protokoll spezifizierten QoS-Parametern, aber auch die Einbettung von QoS-Parametern, die nachträglich von den Betreibern spezifiziert wurden. Bei der Abstraktion der QoS-Technologie werden prinzipielle Unterschiede (Reservierung und Priorisierung) berücksichtigt. Auch bei der Pfadsuche stehen zwei Möglichkeiten zur Verfügung. Einerseits ist eine auf einfache Topologien ausgerichtete Pfadsuche mit gleichzeitiger Reservierung von Ressourcen möglich und andererseits kann bei komplexer oder unbekannter Topologie eine Breitensuche durchgeführt werden, bei der die Ressourcen erst nach der Wahl eines Pfades in einem zweiten Schritt reserviert werden.

Die Konzeption des PPCP ist auf den Einsatz im Internet ausgerichtet, ist aber prinzipiell unabhängig von der Vermittlungsschicht. Die Umsetzung mit IPv4 ist Betreibern vorbehalten, die ausreichend freie Adressen zur Verfügung haben. Die Unabhängigkeit von der Vermittlungsschicht macht aber auch eine Erweiterung der Konzepte möglich, beispielsweise durch den für die Endpunkte transparenten Einsatz von IPv6 in den Transit-Systemen. In diesem Fall würde nicht nur die Empfängeradresse, sondern auch der IP-Header ausgetauscht.

Freie Subnetze vorausgesetzt, ist der Pfadschaltungsteil von PPCP bereit für den Einsatz. Die Erfahrungen mit der prototypischen Implementierung und der Testumgebung sind in die Konzeption eingeflossen und letztere wurde mit einem erfolgreichen Test abgeschlossen. Für die Aushandlung der Dienstgüte eines Pfades sind Mechanismen durch das PPCP verfügbar, die Implementierung und Integration der QoS-Backends stehen aber noch aus.

A. Weitere Szenarien und Kombinationsmöglichkeiten

In diesem Anhang werden einige Szenarien angesprochen, die neue Möglichkeiten und Kombinationen mit anderen Technologien bieten, die aber nicht in die Anforderungen eingeflossen sind.

A.1. Kombination der Pfadschaltung mit „Dynamic DNS“ oder vergleichbaren Verzeichnisdiensten

Die Gateways könnten, anstatt den Endbenutzer oder eine Anwendung über den erfolgreichen Verbindungsaufbau zu informieren, ein DNS-Update ([VTRB 97], auch als DynDNS bekannt) vornehmen, sodass die Verbindung über einen Namen verwendbar ist. Dieser Name kann vor dem Verbindungsaufbau bzw. nach dem Verbindungsabbau auf die eigentliche IP-Adresse des anderen Endes zeigen, sodass die Verbindung für den Benutzer vollkommen transparent ist. Der Nachteil dieser Vorgehensweise ist, dass die IP-Adresse bei bestehender Verbindung vom Standort abhängt. Die Verbindungsadresse ist nicht überall nutzbar oder bringt durch Umwege (Routing in das AS, in dem die Verbindung startet) sogar Nachteile mit sich. Die (Dyn)DNS-Server müssten daher Namen abhängig vom Standort auflösen oder der Name müsste nur von einem Standort aus verwendet werden. Eine Variante dieses Szenarios wäre, die lokale Namensauflösung zu ändern und z. B. bei unixoiden Betriebssystemen einen Eintrag in /etc/hosts zu machen, der die Verbindungsadresse mit einem Namen verknüpft, welcher die Adresse für Anwendungen transparent macht.

A.2. Szenario „Online-Spiele“

Online-Spiele und darunter insbesondere MMORPGs (Massively Multiplayer Online Role-Playing Games) stellen hohe Ansprüche an die Dienstgüte zwischen Client und Server. Die Anbieter begrenzen daher die Anzahl der Spieler pro Server und versuchen durch räumlich nahe Server eine ausreichende Verbindungsqualität zu erreichen. Besonders bei großen In-Game-Events (MMORPGs) oder bei interkontinentalen Spielen (z. B. StarCraft-Matches zwischen Spielern in den USA und Korea) könnten QoS-Verbindungen viel dazu beitragen, das Spielerlebnis zu verbessern. Die Spieleanbieter könnten in ihre Clients die Möglichkeit zum QoS-Verbindungsaufbau integrieren oder, wie im Video-on-Demand-Szenario, den Verbindungsaufbau automatisch bzw. nur zu besonderen Zeiten (Events) durch den Server auslösen. Die Erfüllung der QoS-Parameter könnte wie im VoIP-Szenario funktionieren, echte Garantien für tausende von Spielern wären wahrscheinlich zu aufwändig bzw. zu teuer.

A.3. Standleitung zwischen entfernten Standorten und Kombination mit VPN

Um zwei Firmenstandorte über AS-Grenzen hinweg mit einer virtuellen, QoS-gesicherten Standleitung zu verbinden, kann man die Facetten von VoIP und Videokonferenz mit DynDNS oder VPN verbinden. Aus dem VoIP-Szenario wird das Multiplexing genutzt und aus dem Vidokonferenz-Szenario die Verfügbarkeit und die Garantien. Die Verbindung kann, transparent für die Mitarbeiter, über IPSec

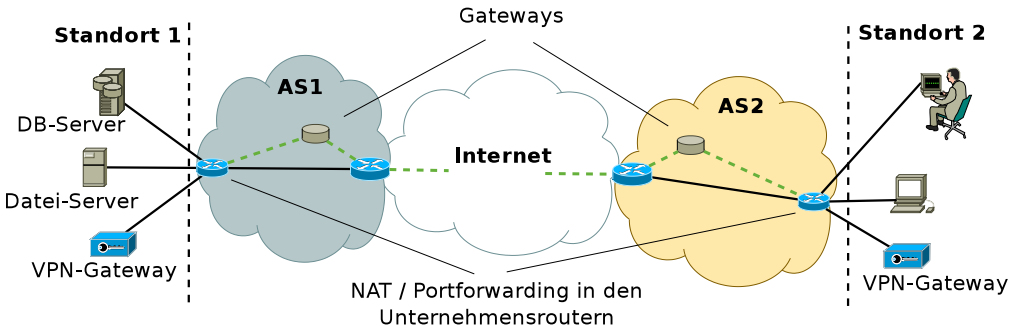


Abbildung A.1.: Virtuelle Standleitung (gestrichelt) eines Unternehmens über das Internet

oder VPN genutzt werden. Alternativ könnte die Verbindung zwischen zwei Routern aufgebaut werden, die in jeweils einem Standort stehen und deren Adresse als Gateway (das „Default-Gateway“ im herkömmlichen Sinn) für das Netz des jeweils anderen Standorts dienen (Abbildung A.1). Eine dritte Möglichkeit besteht darin, verschiedene Dienste des anderen Standorts über die Verbindungsadresse anzusprechen und im anderen Standort über Portweiterleitungen die entsprechenden Server zu erreichen.

A.4. Server-Replikation

Anwendungen, die Echtzeit-Replikation bei Schreibzugriff (z. B. auf Datenbanken oder Datei-Server) über einen QoS-gesicherten Kanal benötigen, könnten das über eine Multicast-Verbindung erreichen. Schreiboperationen würden an eine Adresse (die Verbindungsadresse) gesendet und automatisch per Multicast an zwei Server vermittelt. Die Übertragung der Daten zum Server müsste zuverlässig sein. TCP oder eine Bestätigung an den Client kommt nicht in Frage, da man damit die Transparenz der Replikation aufgibt und zur Verarbeitung der Bestätigungen auch wissen müsste, welche Server beteiligt sind. Es würde reichen, wenn die Zuverlässigkeit der Übertragung ab der Stelle, an der sich die Verbindung aufspaltet, durch die Gateways sichergestellt würde, dann wäre die Antwort eines Servers als Empfangsbestätigung für beide ausreichend. Bleibt diese aus, so werden die Daten nochmal an die Verbindungsadresse, also an beide Server gesendet. Die Server könnten untereinander kommunizieren und vereinbaren, wer die Bestätigung schickt und feststellen, wenn einer der Server ausfällt. Bei Lesezugriffen gäbe es die Möglichkeite, alle Server antworten zu lassen, um (wie beim Flooding) auf jeden Fall die schnellstmögliche Antwort zu erhalten; die später eintreffenden Duplikate würden verworfen.

A.5. Internet-Radio

Viele Radiosender bieten Online-Übertragung an, daher ist hier der Vollständigkeit halber noch Internet-Radio erwähnt. Internet-Radio benötigt, wie Video-on-Demand, eine (gepufferte und relativ geringe) Übertragungsrate. Die Verbindungsdauer hängt aber nicht von der festen Länge eines Films ab, sondern wird durch den Hörer festgelegt. Die Verbindungsdauer variiert dabei von einigen Sekunden („Zappen“) über ein paar Minuten (Nachrichten hören) bis hin zu vielen Stunden (Dauerhören). Wegen der relativ geringen Anforderungen und der starken Varianz der Verbindungsdauer lohnt es sich nicht, automatisch mit dem „Einschalten“ eine Verbindung zu schalten. Sollte das Internet-Radio das Funkradio ablösen und die Anzahl der Clients anwachsen, so wäre Multicast in Verbindung mit QoS aber wieder attraktiv. Auch für das Umschalten auf eine höhere Qualität für Dauerhörer wäre ein Verbindungsaufbau denkbar. Nach einem „Zapp-Timeout“ könnte der Radio-Client eine Verbindung aufbauen und auf eine höhere Qualität umschalten bzw. Mitglied einer Multicast-Verbindung werden.

B. PPCP-PDUs

```
{
 "type": "SEARCH"
 "key":
  {
   "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
   "initiator": "gw1.as1"
# <- comment (not json compliant, just for the prototype)
#  "initiator-time": "2011-12-26T16:43Z"
#  "initiator-signature": "c871191b75f3c1b3faa4de8e7145a176\
#                          5bb1292a1ee4b23667e8fbe203b06398\
#                          87b31ebbbb4a618f5326f422c47844fb
#                          338f059ea3dff94cf2e66a2bd7defc57"
  }
 "receiving-gw": "gw1.as8"
 "sending-gw": "gw1.as2"
 "options": 11
 "metric": 50
 "source": "ende1.as1"
 "sink": "ende3.as7"
 "QoS-Params":
  {
              "latency"  : 50
              "rate"     : 512
              "jitter"   : 5
              "droprate" : 99
              "predefined-set" : "c768e6c6-549c-4333-84ab-36bc8e18e199"
# additional custom qos-pa\-ra\-me\-ter, must be a json object:
              "VoIP settings" :
                {
                  "predefine voip set": "100 connections"
                }
  }
 "sender-time": "2011-12-26T16:43Z"
# "sender-signature": "6222b887dba5df491113d7582878135e\
#                      72a9502fccadc47dbd448f8f1ad4cfa2\
#                      8747da518ce26c2aff19b3cdd401c597\
#                      44f75fa8c53ac37680a6e6bd86ebfc90"
}
```

Listing B.1: SEARCH

```
{
 "type": "SEARCHACK"
 "key":
  {
   "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
   "initiator": "gw1.as1"
  }
 "receiving-gw": "gw1.as2"
 "sending-gw": "gw1.as8"
# timestamp of the confirmed message
 "confirmed-time": "2011-12-26T16:43Z"
}
```

Listing B.2: SEARCHACK

```
{
 "type": "SEARCHFAIL"
 "key":
  {
```

```
    "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
    "initiator": "gw1.as1"
   }
 "receiving-gw": "gw1.as2"
 "sending-gw": "gw1.as8"
 "sender-time": "2011-12-26T17:21Z"
}
```

Listing B.3: SEARCHFAIL

```
{
 "type": "FAILACK"
 "key":
  {
    "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
    "initiator": "gw1.as1"
   }
 "receiving-gw": "gw1.as8"
 "sending-gw": "gw1.as2"
 "confirmed-time": "2011-12-26T17:21Z"
}
```

Listing B.4: FAILACK

```
{
 "Type": "PATHAVAILABLE"
 "key":
  {
    "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
    "initiator": "gw1.as1"
   }
 "receiving-gw": "gw1.as2"
 "sending-gw": "gw1.as8"
 "metric": 30
# source and sink only necessary for multicast
# "source": "ende1.as1"
# "sink": "ende3.as7"
 "sender-time": "2011-12-26T19:21Z"
}
```

Listing B.5: PATHAVAILABLE

```
{
 "type": "PATHACK"
 "key":
  {
    "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
    "initiator": "gw1.as1"
   }
 "receiving-gw": "gw1.as8"
 "sending-gw": "gw1.as2"
 "confirmed-time": "2011-12-26T19:21Z"
}
```

Listing B.6: PATHACK

```
{
 "Type": "CONNECT"
 "key":
  {
    "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
    "initiator": "gw1.as1"
#   "initiator-time": "2011-12-26T20:04Z"
#   "initiator-signature": "c871191b75f3c1b3faa4de8e7145a176\
#                           5bb1292a1ee4b23667e8fbe203b06398\
#                           87b31ebbbb4a618f5326f422c47844fb
#                           338f059ea3dff94cf2e66a2bd7defc57"
   }
 "receiving-gw": "gw1.as8"
```

```
 "sending-gw": "gw1.as2"
 "rewrite-addr": "10.0.12.199"
 "keepalive-timeout": 999
 "sender-time": "2011-12-26T20:08Z"

# only needed for a search on the fly:
# "metric": 30
# "source": "ende1.as1"
# "sink": "ende3.as7"
# "options": 11
# "source": "ende1.as1"
# "sink": "ende3.as7"
# "QoS-Params":
#  {
#              "latency"  : 50
#              "rate"     : 512
#              "jitter"   : 5
#              "droprate" : 99
#              "predefined-set" : "c768e6c6-549c-4333-84ab-36bc8e18e199"
}
```

Listing B.7: CONNECT

```
{
 "type": "CONNECTACK"
 "key":
  {
   "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
   "initiator": "gw1.as1"
  }
 "receiving-gw": "gw1.as2"
 "sending-gw": "gw1.as8"
 "confirmed-time": "2011-12-26T20:08Z"
}
```

Listing B.8: CONNECTACK

```
{
 "type": "CONNECTFAIL"
 "key":
  {
   "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
   "initiator": "gw1.as1"
  }
 "receiving-gw": "gw1.as2"
 "sending-gw": "gw1.as8"
 "sender-time": "2011-12-26T20:08Z"
}
```

Listing B.9: CONNECTFAIL

```
{
 "Type": "ESTABLISHED"
 "key":
  {
   "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
   "initiator": "gw1.as1"
  }
 "receiving-gw": "gw1.as8"
 "sending-gw": "gw1.as2"
# only for multicast:
# "source": "ende1.as1"
# "sink": "ende3.as7"
 "rewrite-addr" : "10.8.230.33"
 "sender-time": "2011-12-26T24:00Z"
}
```

Listing B.10: ESTABLISHED

```
{
 "type": "ESTABLISHEDACK"
 "key":
  {
   "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
   "initiator": "gw1.as1"
  }
 "receiving-gw": "gw1.as2"
 "sending-gw": "gw1.as8"
 "confirmed-time": "2011-12-26T24:00Z"
}
```

Listing B.11: ESTABLISHEDACK

```
{
 "type": "SEARCHCANCEL"
 "key":
  {
   "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
   "initiator": "gw1.as1"
  }
 "receiving-gw": "gw1.as4"
 "sending-gw": "gw1.as2"
 "sender-time": "2011-12-26T24:01Z"
}
```

Listing B.12: SEARCHCANCEL

```
{
 "type": "CANCELACK"
 "key":
  {
   "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
   "initiator": "gw1.as1"
  }
 "receiving-gw": "gw1.as2"
 "sending-gw": "gw1.as4"
 "confirmed-time": "2011-12-26T24:01Z"
}
```

Listing B.13: CANCELACK

```
{
 "Type": "KEEPALIVE"
 "key":
  {
   "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
   "initiator": "gw1.as1"
  }
 "receiving-gw": "gw1.as8"
 "sending-gw": "gw1.as2"
 "sender-time": "2011-12-26T34:01Z"
 "keepalive": 1200
}
```

Listing B.14: KEEPALIVE

```
{
 "Type": "KEEPALIVEACK"
 "key":
  {
   "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
   "initiator": "gw1.as1"
  }
 "receiving-gw": "gw1.as2"
 "sending-gw": "gw1.as8"
 "confirmed-time": "2011-12-26T34:01Z"
}
```

Listing B.15: KEEPALIVEACK

```
{
 "Type": "ERROR"
 "key":
  {
   "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
   "initiator": "gw1.as1"
  }
 "receiving-gw": "gw1.as8"
 "sending-gw": "gw1.as2"
 "sender-time": "2011-12-26T39:02Z"
 "errno" = -1
 "log-message": "uh oh! that didn't work out..."
}
```

Listing B.16: ERROR

```
{
 "type": "ERRORACK"
 "key":
  {
   "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
   "initiator": "gw1.as1"
  }
 "receiving-gw": "gw1.as2"
 "sending-gw": "gw1.as4"
 "confirmed-time": "2011-12-26T39:02Z"
}
```

Listing B.17: ERRORACK

```
{
 "Type": "REDIRECT"
 "key":
  {
   "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
   "initiator": "gw1.as1"
  }
 "receiving-gw": "gw1.as1"
 "sending-gw": "gw1.as2"
 "next-gateway": "gw1.as8"
 "alternative-gateway": "gw1.as4"
 "sender-time": "2011-12-26T50:50Z"
}
```

Listing B.18: REDIRECT

```
{
 "type": "REDIRECTACK"
 "key":
  {
   "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
   "initiator": "gw1.as1"
  }
 "receiving-gw": "gw1.as2"
 "sending-gw": "gw1.as1"
 "confirmed-time": "2011-12-26T50:50Z"
}
```

Listing B.19: REDIRECTACK

```
{
 "type": "DISCONNECT"
 "key":
  {
   "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
   "initiator": "gw1.as1"
  }
 "receiving-gw": "gw1.as8"
 "sending-gw": "gw1.as4"
```

```
10  "sender-time": "2011-12-26T59:02Z"
11 }
```

Listing B.20: DISCONNECT

```
1  {
2   "type": "DISCONNECTACK"
3   "key":
4    {
5     "id":"fd0b376c-1053-4e3e-ad49-7fc3d5e43fef"
6     "initiator": "gw1.as1"
7    }
8   "receiving-gw": "gw1.as4"
9   "sending-gw": "gw1.as8"
10  "confirmed-time": "2011-12-26T59:02Z"
11 }
```

Listing B.21: DISCONNECTACK

C. Konfigurationsdateien und Skripte

Listing C.1: Die ppcpd-Konfigurationsdatei auf gw1.as2

```
# logmask, define log-level as sum of
# default: 1
# verbose: 2
# protocol: 4
# protocol parser 8
# signalling: 16
# network: 32
# network dump: 64
# debug: 128 (logmask >= 128 enables debug logging options)
# protocol debugging: 256
# protocol parser debugging: 512
# signal debugging: 1024
# qos debugging: 2048
# qos parser debugging: 4096
# gw handler debugging: 8192
# log everything: 65353
logmask 1 + 2 + 8192 +64 + 128 +256
# logfile:
logfile stdout
# working directory (trailing slash necessary):
workdir /var/run/ppcpd/
# force ppcp communication over ip version 4, 6 or 0 (auto):
force_ip_version 4
# listen on ppcpdip for ppcp communication, 0.0.0.0 for all available addresses:
ppcpdip 0.0.0.0
# read ppcp datagrams from UDP-Port, default 7429
ppcpdport 7429
# hostname, used as sender addres in ppcp
hostname gw1.as2

# connect to rewrited on host
rewritedaddr localhost
# rewrited listening on TCP-Port
rewritedport 7427
# QoS subnet
rewritenet 10.0.12.192/26

# file that associates networks with (neighbour) gws
gwfile /etc/ppcpd.gws
```

Listing C.2: Debian-Konfigurationsdatei /etc/network/interfaces auf gw1.as2

```
auto lo
iface lo inet loopback

manual eth0
iface eth0 inet static
        address 172.20.2.51
        netmask 255.255.0.0
        broadcast 172.20.255.255
        gateway 172.20.255.254

auto eth1
iface eth1 inet static
        address 10.0.12.254
        netmask 255.255.255.0
        broadcast 10.0.12.255
```

Listing C.3: Die Tabelle für den Gateway-Lookup auf gw1.as2

```
# This file associates networks with ppcpd-gateways.
# each line contains a network and one ppcp-gateway,
# that could serve as the next (ppcp) hop.
# for each network, multiple lines are allowed,
# the topmost entry will be treated with the
# highest priority

# Domain as0 (QoS-subnet: 10.0.10.192/26):
10.0.10.0/24 gw1.as1
10.0.10.0/24 gw1.as4

# Domain as1 (QoS-subnet: 10.0.11.192/26):
10.0.11.0/24 gw1.as1

# Domain as2 (QoS-subnet: 10.0.12.192/26):
10.0.12.0/24 gw1.as2

# domain as3 (QoS-subnet: 10.3.96.0/20):
10.3.0.0/16 gw1.as1
10.3.0.0/16 gw1.as4

# Domain as4 (QoS-subnet: 10.4.255.0/24):
10.4.0.0/16 gw1.as4

# Domain as7 (QoS-subnet: 10.7.230.0/24):
10.8.0.0/16 gw1.as8

# Domain as8 (QoS-subnet: 10.8.230.0/24):
10.8.0.0/16 gw1.as8
```

Listing C.4: Debian-Start-Skript (/etc/network/if-up.d/routing.sh) für die Konfiguration des Routing gw1.as2

```
#!/bin/sh
if [ "$IFACE" = eth1 ]; then
    route add -net 10.0.11.0 netmask 255.255.255.0 gw rt1.as2 dev eth1
    ip link set eth2 up
    route add -host rt1.as4 dev eth2
    route add -host rt1.as8 dev eth2
    route add -net 10.4.0.0 netmask 255.255.0.0 gw rt1.as4 dev eth2
    route add -net 10.8.0.0 netmask 255.255.0.0 gw rt1.as8 dev eth2
    route add -net 10.7.0.0 netmask 255.255.0.0 gw rt1.as8 dev eth2
    route add default gw rt1.as4 dev eth2
    echo 1 > /proc/sys/net/ipv4/ip_forward
    exit 0
else
    exit 0
fi
```

Listing C.5: Debian-Stop-Skript (/etc/network/if-down.d/routing.sh) für die Konfiguration des Routing auf gw1.as2

```
#!/bin/sh
if [ "$IFACE" = eth1 ]; then
    echo 0 > /proc/sys/net/ipv4/ip_forward
    route del default gw rt1.as4 dev eth2
    route del -net 10.0.11.0 netmask 255.255.255.0 gw rt1.as2 dev eth1
    route del -net 10.4.0.0 netmask 255.255.0.0 gw rt1.as4 dev eth2
    route del -net 10.8.0.0 netmask 255.255.0.0 gw rt1.as8 dev eth2
    route del -net 10.7.0.0 netmask 255.255.0.0 gw rt1.as8 dev eth2
    route del -host rt1.as4 dev eth2
    route del -host rt1.as8 dev eth2
    ip link set eth2 down
    exit 0
else
    exit 0
fi
```

Listing C.6: Skript zur Erzeugung der Bridges für das Testnetz

```
#!/bin/bash
### bridges for the uml net ###
if [ "$1" == "uml" ]; then
    brctl addbr br0_as0
    brctl addbr br0_as0_as1
    brctl addbr br0_as0_as3
    brctl addbr br0_as1
    brctl addbr br0_as1_as2
    brctl addbr br0_as1_as3
    brctl addbr br0_as1_as4
    brctl addbr br0_as2
    brctl addbr br0_as3
    brctl addbr br1_as3
    brctl addbr br2_as3
    brctl addbr br3_as3
    brctl addbr br0_as3_as4
    brctl addbr br0_as4
    brctl addbr br1_as4
    brctl addbr br2_as4
    brctl addbr br3_as4
    brctl addbr br4_as4
    brctl addbr br0_as2_as4_as8

    brctl addbr br0_mn

#########################################################
    brctl stp br0_as0         on
    brctl stp br0_as0_as1     on
    brctl stp br0_as0_as3     on
    brctl stp br0_as1         on
    brctl stp br0_as1_as2     on
    brctl stp br0_as1_as3     on
    brctl stp br0_as1_as4     on
    brctl stp br0_as2         on
    brctl stp br0_as3         on
    brctl stp br1_as3         on
    brctl stp br2_as3         on
    brctl stp br3_as3         on
    brctl stp br0_as3_as4     on
    brctl stp br0_as4         on
    brctl stp br1_as4         on
    brctl stp br2_as4         on
    brctl stp br3_as4         on
    brctl stp br4_as4         on
    brctl stp br0_as2_as4_as8 on

    brctl stp br0_mn          on

#########################################################
    ifconfig br0_as0         hw ether 02:b0:a0:00:00:00 up
    ifconfig br0_as0_as1     hw ether 02:b0:a0:a1:00:00 up
    ifconfig br0_as0_as3     hw ether 02:b0:a0:a3:00:00 up
    ifconfig br0_as1         hw ether 02:b0:a1:00:00:00 up
    ifconfig br0_as1_as2     hw ether 02:b0:a1:a2:00:00 up
    ifconfig br0_as1_as3     hw ether 02:b0:a1:a3:00:00 up
    ifconfig br0_as1_as4     hw ether 02:b0:a1:a4:00:00 up
    ifconfig br0_as2         hw ether 02:b0:a2:00:00:00 up
    ifconfig br0_as3         hw ether 02:b0:a3:00:00:00 up
    ifconfig br1_as3         hw ether 02:b1:a3:00:00:00 up
    ifconfig br2_as3         hw ether 02:b2:a3:00:00:00 up
    ifconfig br3_as3         hw ether 02:b3:a3:00:00:00 up
    ifconfig br0_as3_as4     hw ether 02:b0:a3:a4:00:00 up
    ifconfig br0_as4         hw ether 02:b0:a4:00:00:00 up
    ifconfig br1_as4         hw ether 02:b1:a4:00:00:00 up
    ifconfig br2_as4         hw ether 02:b2:a4:00:00:00 up
    ifconfig br3_as4         hw ether 02:b3:a4:00:00:00 up
    ifconfig br4_as4         hw ether 02:b4:a4:00:00:00 up
    ifconfig br0_mn          hw ether 02:b0:af:00:00:00 up
    ifconfig br0_as2_as4_as8 hw ether 02:b0:a2:a4:a8:00 up
```

```
70
71  ########################################################
72      brctl addif br0_as2_as4_as8 eth0
73
74  elif [ "$1" == "xen" ]; then
75  ### bridges for the xen net ###
76      brctl addbr br0_as7
77      brctl addbr br1_as7
78      brctl addbr br2_as7
79      brctl addbr br0_as7_as8
80      brctl addbr br0_as8
81      brctl addbr br1_as8
82      brctl addbr br2_as8
83      brctl addbr br0_as2_as4_as8
84
85  ############################################################
86      brctl stp br0_as7         on
87      brctl stp br1_as7         on
88      brctl stp br2_as7         on
89      brctl stp br0_as7_as8     on
90      brctl stp br0_as8         on
91      brctl stp br1_as8         on
92      brctl stp br2_as8         on
93      brctl stp br0_as2_as4_as8 on
94
95  ############################################################
96      ifconfig br0_as7         hw ether 02:b0:a7:00:00:00 up
97      ifconfig br1_as7         hw ether 02:b1:a7:00:00:00 up
98      ifconfig br2_as7         hw ether 02:b2:a7:00:00:00 up
99      ifconfig br0_as7_as8     hw ether 02:b0:a7:a8:00:00 up
100     ifconfig br0_as8         hw ether 02:b0:a8:00:00:00 up
101     ifconfig br1_as8         hw ether 02:b1:a8:00:00:00 up
102     ifconfig br2_as8         hw ether 02:b2:a8:00:00:00 up
103     ifconfig br0_as2_as4_as8 hw ether 02:b0:a2:a4:a8:01 up
104
105 ############################################################
106     brctl addif br0_as2_as4_as8 eth1
107 else
108     echo "usage: $0 (xen|uml)"
109 fi
```

Listing C.7: Skript für die hostseitige Verbindung des Testnetzes mit dem Internet

```
1  #!/bin/sh
2  extiface=wlan0
3  brctl addbr br0
4  ifconfig br0 172.20.255.1 netmask 255.255.0.0 up
5  route add  -host 10.3.255.254 dev br0
6  route add -net 10.0.0.0 netmask 255.0.0.0 gw 10.3.255.254
7
8  /etc/init.d/bind9 start
9  echo 1 > /proc/sys/net/ipv4/ip_forward
10 iptables -t nat -A POSTROUTING -o $extiface -j MASQUERADE
11 iptables -A FORWARD -t filter -o $extiface -m state \
12     --state NEW,ESTABLISHED,RELATED -j ACCEPT
13
14 iptables -A FORWARD -t filter -i $extiface -m state \
15     --state ESTABLISHED,RELATED -j ACCEPT
16
17 exit 0
```

Listing C.8: „bash-Datenkbank“ für die Interfaces in net-up und start-uml

```
1  # MAC addresses (e|c|d) for end|gw|router
2  # K,N,M for host-, as- and device-number
3  # e for ethernet  00:(e|c|d)K:aN:eM:00:00
4
5  export ende1_as0_eth0_HWADR=02:e1:a0:e0:00:00
6  export ende1_as0_eth1_HWADR=02:e1:a0:e1:00:00
```

```
7
8  export gw1_as0_eth0_HWADR=02:c1:a0:e0:00:00
9  export gw1_as0_eth1_HWADR=02:c1:a0:e1:00:00
10 export gw1_as0_eth2_HWADR=02:c1:a0:e2:00:00
11 export gw1_as0_eth3_HWADR=02:c1:a0:e3:00:00
12
13 export gw2_as0_eth0_HWADR=02:c2:a0:e0:00:00
14 export gw2_as0_eth1_HWADR=02:c2:a0:e1:00:00
15 export gw2_as0_eth2_HWADR=02:c2:a0:e2:00:00
16
17 export ende1_as1_eth0_HWADR=02:e1:a1:e0:00:00
18 export ende1_as1_eth1_HWADR=02:e1:a1:e1:00:00
19
20 export rt1_as1_eth0_HWADR=02:d1:a1:e0:00:00
21 export rt1_as1_eth1_HWADR=02:d1:a1:e1:00:00
22 export rt1_as1_eth2_HWADR=02:d1:a1:e2:00:00
23 export rt1_as1_eth3_HWADR=02:d1:a1:e3:00:00
24 export rt1_as1_eth4_HWADR=02:d1:a1:e4:00:00
25
26 export gw1_as1_eth0_HWADR=02:c1:a1:e0:00:00
27 export gw1_as1_eth1_HWADR=02:c1:a1:e1:00:00
28 export gw1_as1_eth2_HWADR=02:c1:a1:e2:00:00
29
30 export gw1_as2_eth0_HWADR=02:c1:a2:e0:00:00
31 export gw1_as2_eth1_HWADR=02:c1:a2:e1:00:00
32 export gw1_as2_eth2_HWADR=02:c1:a2:e2:00:00
33
34 export rt1_as2_eth0_HWADR=02:d1:a2:e0:00:00
35 export rt1_as2_eth1_HWADR=02:d1:a2:e1:00:00
36 export rt1_as2_eth2_HWADR=02:d1:a2:e2:00:00
37
38 export ende1_as3_eth0_HWADR=02:e1:a3:e0:00:00
39 export ende1_as3_eth1_HWADR=02:e1:a3:e1:00:00
40
41 export gw1_as3_eth0_HWADR=02:c1:a3:e0:00:00
42 export gw1_as3_eth1_HWADR=02:c1:a3:e1:00:00
43 export gw1_as3_eth2_HWADR=02:c1:a3:e2:00:00
44
45 export gw2_as3_eth0_HWADR=02:d2:a3:e0:00:00
46 export gw2_as3_eth1_HWADR=02:d2:a3:e1:00:00
47 export gw2_as3_eth2_HWADR=02:d2:a3:e2:00:00
48
49 export rt1_as3_eth0_HWADR=02:d1:a3:e0:00:00
50 export rt1_as3_eth1_HWADR=02:d1:a3:e1:00:00
51 export rt1_as3_eth2_HWADR=02:d1:a3:e2:00:00
52 export rt1_as3_eth3_HWADR=02:d1:a3:e3:00:00
53 export rt1_as3_eth4_HWADR=02:d1:a3:e4:00:00
54 export rt1_as3_eth4_HWADR=02:d1:a3:e5:00:00
55
56 export rt2_as3_eth0_HWADR=02:c2:a3:e0:00:00
57 export rt2_as3_eth1_HWADR=02:c2:a3:e1:00:00
58 export rt2_as3_eth2_HWADR=02:c2:a3:e2:00:00
59
60 export rt1_as4_eth0_HWADR=02:d1:a4:e0:00:00
61 export rt1_as4_eth1_HWADR=02:d1:a4:e1:00:00
62 export rt1_as4_eth2_HWADR=02:d1:a4:e2:00:00
63 export rt1_as4_eth3_HWADR=02:d1:a4:e3:00:00
64 export rt1_as4_eth3_HWADR=02:d1:a4:e4:00:00
65
66 export rt2_as4_eth0_HWADR=02:d2:a4:e0:00:00
67 export rt2_as4_eth1_HWADR=02:d2:a4:e1:00:00
68 export rt2_as4_eth2_HWADR=02:d2:a4:e2:00:00
69 export rt2_as4_eth3_HWADR=02:d2:a4:e3:00:00
70 export rt2_as4_eth4_HWADR=02:d2:a4:e4:00:00
71
72 export rt3_as4_eth0_HWADR=02:c3:a4:e0:00:00
73 export rt3_as4_eth1_HWADR=02:c3:a4:e1:00:00
74 export rt3_as4_eth2_HWADR=02:c3:a4:e2:00:00
75
76 export gw1_as4_eth0_HWADR=02:c1:a4:e0:00:00
```

```
77 export gw1_as4_eth1_HWADR=02:c1:a4:e1:00:00
78 export gw1_as4_eth2_HWADR=02:c1:a4:e2:00:00
79
80 export ende1_as4_eth0_HWADR=02:e1:a4:e0:00:00
81 export ende1_as4_eth1_HWADR=02:e1:a4:e1:00:00
82
83 export ende2_as4_eth0_HWADR=02:e2:a4:e0:00:00
84 export ende2_as4_eth1_HWADR=02:e2:a4:e1:00:00
85
86 export mn_asmn_eth0_HWADR=02:ff:af:e0:00:00
87 export mn_asmn_eth1_HWADR=02:ff:af:e1:00:00
88
89
90 # Bridge-Connections
91 export ende1_as0_eth0_br=br0
92 export ende1_as0_eth1_br=br0_as0
93
94 export gw1_as0_eth0_br=br0
95 export gw1_as0_eth1_br=br0_as0
96 export gw1_as0_eth2_br=br0_as0_as3
97 #export gw1_as0_eth3_br=br0_as0
98
99 export gw2_as0_eth0_br=br0
100 export gw2_as0_eth1_br=br0_as0
101 export gw2_as0_eth2_br=br0_as0_as1
102
103 export ende1_as1_eth0_br=br0
104 export ende1_as1_eth1_br=br0_as1
105
106 export rt1_as1_eth0_br=br0
107 export rt1_as1_eth1_br=br0_as1
108 export rt1_as1_eth2_br=br0_as1_as3
109 export rt1_as1_eth3_br=br0_as1_as4
110 export rt1_as1_eth4_br=br0_as1_as2
111
112 export gw1_as1_eth0_br=br0
113 export gw1_as1_eth1_br=br0_as1
114 export gw1_as1_eth2_br=br0_as0_as1
115
116 export rt1_as2_eth0_br=br0
117 export rt1_as2_eth1_br=br0_as2
118 export rt1_as2_eth2_br=br0_as1_as2
119
120 export gw1_as2_eth0_br=br0
121 export gw1_as2_eth1_br=br0_as2
122 export gw1_as2_eth2_br=br0_as2_as4_as8
123
124 export ende1_as3_eth0_br=br0
125 export ende1_as3_eth1_br=br0_as3
126
127 export gw1_as3_eth0_br=br0
128 export gw1_as3_eth1_br=br1_as3
129 export gw1_as3_eth2_br=br0_as3_as4
130
131 export gw2_as3_eth0_br=br0
132 export gw2_as3_eth1_br=br2_as3
133 export gw2_as3_eth2_br=br0_as1_as3
134
135 export rt1_as3_eth0_br=br0
136 export rt1_as3_eth1_br=br0_as3
137 export rt1_as3_eth2_br=br1_as3
138 export rt1_as3_eth3_br=br2_as3
139 export rt1_as3_eth4_br=br3_as3
140 export rt1_as3_eth5_br=br0_mn
141
142 export rt2_as3_eth0_br=br0
143 export rt2_as3_eth1_br=br3_as3
144 export rt2_as3_eth2_br=br0_as0_as3
145
146 export rt1_as4_eth0_br=br0
```

```
export rt1_as4_eth1_br=br3_as4
export rt1_as4_eth2_br=br0_as2_as4_as8
export rt1_as4_eth3_br=br0_as1_as4
export rt1_as4_eth4_br=br0_as2_as4_as8

export rt2_as4_eth0_br=br0
export rt2_as4_eth1_br=br3_as4
export rt2_as4_eth2_br=br4_as4
export rt2_as4_eth3_br=br1_as4
export rt2_as4_eth4_br=br2_as4

export rt3_as4_eth0_br=br0
export rt3_as4_eth1_br=br4_as4
export rt3_as4_eth2_br=br0_as3_as4

export gw1_as4_eth0_br=br0
export gw1_as4_eth1_br=br0_as4
export gw1_as4_eth2_br=br1_as4

export ende1_as4_eth0_br=br0
export ende1_as4_eth1_br=br2_as4

export ende2_as4_eth0_br=br0
export ende2_as4_eth1_br=br0_as4

export mn_asmn_eth0_br=br0
export mn_asmn_eth1_br=br0_mn

#export mn_asmn_eth0_ADR=10.10.10.210
#export mn_asmn_eth0_MASK=255.255.0.0
```

Listing C.9: Skript zur Erzeugung der TAP-Schnittstellen für die UML-Instanzen

```
#!/bin/bash
export ARG1=$(echo $1|sed 's/\./_/')
# Adressliste
source $(dirname $0)/interface_database
BRTMP=${ARG1}_br
BRIDGE=${!BRTMP}
IFACE=${ARG1}
IFACEADR=${ARG1}_ADR
IFACEHWADRTMP=${ARG1}_HWADR
IFACEHWADR=${!IFACEHWADRTMP}
IFACEMASK=${ARG1}_MASK

macchanger -m $IFACEHWADR $IFACE
ip link set $IFACE up
brctl addif $BRIDGE $IFACE
```

Listing C.10: Skript zum Starten einer UML-Instanz

```
#!/bin/bash
source $(dirname $0)/interfaces
NAME=$2
IFNAME=$(echo $NAME|sed 's/\./_/')
OWNER=$3
DIR=/home/files/uml_images/versuchsnetz/as$1/$2
IFACESNUM=1

if [ ! $UMLMEM ]; then
    UMLMEM=96m
fi

if [ $4 ]; then
    IFACESNUM=$4
fi

if [[ ! $5 && ( $3 || $4 ) ]] ; then
    echo starting $NAME
```

```
19
20      for A in $(seq 0 $(($IFACESNUM -1 ))); do
21   su -c "tunctl -u $OWNER -t $IFNAME\_eth$A"
22   $(dirname $0)/net-up $IFNAME\_eth$A
23   HWADRTMP=$IFNAME\_eth$A\_HWADR
24   HWADR=$(echo ${!HWADRTMP}|sed 's/00:00/00:11/')
25   echo $HWADR
26   export IFACES="$IFACES eth$A=tuntap,$IFNAME\_eth$A,$HWADR"
27      done
28      UML_CMD="/usr/bin/linux.uml umid=$NAME uml_dir=$DIR mem=$UMLMEM \
29        ubd0=$DIR/$NAME.rootfs  ubd1=$DIR/$NAME.swap con0=fd:0,fd:1 con=null $IFACES"
30  # NDETACH is broken..
31      if [ $NDETACH ]; then
32   echo "Starting UML in non-detached mode ..."
33   su - $OWNER -c "$UML_CMD"
34      else
35   echo "Starting UML in detached mode ..."
36   su - $OWNER -c "$UML_CMD" &> $DIR/$NAME.log
37      fi
38
39      for A in $(seq 0 $(($IFACESNUM -1))); do
40   $(dirname $0)/net-down $IFNAME\_eth$A
41   su -c "tunctl -d $IFNAME\_eth$A"
42      done
43      echo done with uml-instance: $NAME
44
45  else
46      echo usage: $0 AS_number machine_name user [number_of_network_interfaces]
47      echo possible variables: UMLMEM (defaults to 96m), NDETACH
48  fi
```

Listing C.11: Skript zum Starten des Versuchsnetzes (User-Mode-Linux)

```
1  #!/bin/bash
2  UMLOWNER=dafeu
3  UMLSTARTSCRIPT=$(dirname $0)/start-uml
4
5  sudo $(dirname $0)/bridge2internet uml
6  sudo $(dirname $0)/create_bridges uml
7  sudo screen -d -m -S ende1.as0 $UMLSTARTSCRIPT 0 ende1.as0 $UMLOWNER 2
8  sudo screen -d -m -S gw1.as0   $UMLSTARTSCRIPT 0 gw1.as0   $UMLOWNER 3
9  sudo screen -d -m -S gw2.as0   $UMLSTARTSCRIPT 0 gw2.as0   $UMLOWNER 3
10 sudo screen -d -m -S gw1.as1   $UMLSTARTSCRIPT 1 gw1.as1   $UMLOWNER 3
11 sudo screen -d -m -S rt1.as1   $UMLSTARTSCRIPT 1 rt1.as1   $UMLOWNER 5
12 sudo screen -d -m -S ende1.as1 $UMLSTARTSCRIPT 1 ende1.as1 $UMLOWNER 2
13 sudo screen -d -m -S gw1.as2   $UMLSTARTSCRIPT 2 gw1.as2   $UMLOWNER 3
14 sudo screen -d -m -S rt1.as2   $UMLSTARTSCRIPT 2 rt1.as2   $UMLOWNER 3
15 sudo screen -d -m -S gw1.as3   $UMLSTARTSCRIPT 3 gw1.as3   $UMLOWNER 3
16 sudo screen -d -m -S gw2.as3   $UMLSTARTSCRIPT 3 gw2.as3   $UMLOWNER 3
17 sudo screen -d -m -S rt1.as3   $UMLSTARTSCRIPT 3 rt1.as3   $UMLOWNER 6
18 sudo screen -d -m -S rt2.as3   $UMLSTARTSCRIPT 3 rt2.as3   $UMLOWNER 3
19 sudo screen -d -m -S ende1.as3 $UMLSTARTSCRIPT 3 ende1.as3 $UMLOWNER 2
20 sudo screen -d -m -S rt1.as4   $UMLSTARTSCRIPT 4 rt1.as4   $UMLOWNER 5
21 sudo screen -d -m -S rt2.as4   $UMLSTARTSCRIPT 4 rt2.as4   $UMLOWNER 5
22 sudo screen -d -m -S rt3.as4   $UMLSTARTSCRIPT 4 rt3.as4   $UMLOWNER 3
23 sudo screen -d -m -S gw1.as4   $UMLSTARTSCRIPT 4 gw1.as4   $UMLOWNER 3
24 sudo screen -d -m -S ende1.as4 $UMLSTARTSCRIPT 4 ende1.as4 $UMLOWNER 2
25 sudo screen -d -m -S ende2.as4 $UMLSTARTSCRIPT 4 ende2.as4 $UMLOWNER 2
26
27 sudo screen -d -m -S mn.asmn   $UMLSTARTSCRIPT mn mn.asmn  $UMLOWNER 2
```

Literaturverzeichnis

[AAC+ 03] Akyildiz, Ian F., Tricha Anjali, Leonardo C. Chen, Jaudelice Cavalcante de Oliveira, Caterina M. Scoglio, Agatino Sciuto, Jeffrey A. Smith und George Uhl: *A new traffic engineering manager for DiffServ/MPLS networks: design and implementation on an IP QoS Testbed.* Computer Communications, 26(4):388–403, 2003, http://dblp.uni-trier.de/db/journals/comcom/comcom26.html#AkyildizACOSSSU03 . 25

[ABG+ 01] Awduche, D., L. Berger, D. Gan, T. Li, V. Srinivasan und G. Swallow: *RSVP-TE: Extensions to RSVP for LSP Tunnels.* RFC 3209 (Proposed Standard), Dezember 2001, http://www.ietf.org/rfc/rfc3209.txt . Updated by RFCs 3936, 4420, 4874, 5151, 5420, 5711. 1, 25

[BaSc 04] Baset, Salman und Henning Schulzrinne: *An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol.* CoRR, abs/cs/0412017, 2004. 27

[BBC+ 98] Blake, S., D. Black, M. Carlson, E. Davies, Z. Wang und W. Weiss: *An Architecture for Differentiated Service.* RFC 2475 (Informational), Dezember 1998, http://www.ietf.org/rfc/rfc2475.txt . Updated by RFC 3260. 25

[BCS 94] Braden, R., D. Clark und S. Shenker: *Integrated Services in the Internet Architecture: an Overview.* RFC 1633 (Informational), Juni 1994, http://www.ietf.org/rfc/rfc1633.txt . 1

[Brad 89] Braden, R.: *Requirements for Internet Hosts - Communication Layers.* RFC 1122 (Standard), Oktober 1989, http://www.ietf.org/rfc/rfc1122.txt . Updated by RFCs 1349, 4379, 5884, 6093, 6298. 26

[BZB+ 97] Braden, R., L. Zhang, S. Berson, S. Herzog und S. Jamin: *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification.* RFC 2205 (Proposed Standard), September 1997, http://www.ietf.org/rfc/rfc2205.txt . Updated by RFCs 2750, 3936, 4495, 5946, 6437. 1, 25

[BZKS 08] Brenner, Walter, Rüdiger Zarnekow, J. Kruse und A. Sidler: *Qualität im Internet.* Elektrotechnik und Informationstechnik, 125(7-8):268–273, 2008. 26

[CNRS 98] Crawley, E., R. Nair, B. Rajagopalan und H. Sandick: *A Framework for QoS-based Routing in the Internet.* RFC 2386 (Informational), August 1998, http://www.ietf.org/rfc/rfc2386.txt . 25, 26

[DeCh 02] Demichelis, C. und P. Chimento: *IP Packet Delay Variation Metric for IP Performance Metrics (IPPM).* RFC 3393 (Proposed Standard), November 2002, http://www.ietf.org/rfc/rfc3393.txt . 43

[DKMY 11] Danciu, V., D. Kranzlmüller, M. Metzker und M. Yampolskiy: *A Border–Friendly, Non–Overlay Mechanism for Inter–Domain QoS Support in the Internet.* In: *Proceedings of the ITI 2011 33rd International Conference on Information Technology Interfaces*, Band 2011, Seiten 97–102, Zagreb, Croatia, Juni 2011. University of Zagreb University Computing Centre. iii, 2, 12, 25

[HaBa 96] Hawkinson, J. und T. Bates: *Guidelines for creation, selection, and registration of an Autonomous System (AS).* RFC 1930 (Best Current Practice), März 1996, http://www.ietf.org/rfc/rfc1930.txt . 3

[JAC+ 02] Jamoussi, B., L. Andersson, R. Callon, R. Dantu, L. Wu, P. Doolan, T. Worster, N. Feldman, A. Fredette, M. Girish, E. Gray, J. Heinanen, T. Kilty und A. Malis: *Constraint-Based LSP Setup using LDP*. RFC 3212 (Proposed Standard), Januar 2002, http://www.ietf.org/rfc/rfc3212.txt . Updated by RFC 3468. 25

[Jaco 88] Jacobson, V.: *Congestion avoidance and control*. SIGCOMM Comput. Commun. Rev., 18:314–329, August 1988, http://doi.acm.org/10.1145/52325.52356 . 26

[KeSe 05] Kent, S. und K. Seo: *Security Architecture for the Internet Protocol*. RFC 4301 (Proposed Standard), Dezember 2005, http://www.ietf.org/rfc/rfc4301.txt . Updated by RFC 6040. 16

[LGL+ 96] Leech, M., M. Ganis, Y. Lee, R. Kuris, D. Koblas und L. Jones: *SOCKS Protocol Version 5*. RFC 1928 (Proposed Standard), März 1996, http://www.ietf.org/rfc/rfc1928.txt . 26

[NBBB 98] Nichols, K., S. Blake, F. Baker und D. Black: *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*. RFC 2474 (Proposed Standard), Dezember 1998, http://www.ietf.org/rfc/rfc2474.txt . Updated by RFCs 3168, 3260. 10, 25, 26

[Post 81] Postel, J.: *Internet Protocol*. RFC 791 (Standard), September 1981, http://www.ietf.org/rfc/rfc791.txt . Updated by RFC 1349. 25, 26

[ReLi 95] Rekhter, Y. und T. Li: *A Border Gateway Protocol 4 (BGP-4)*. RFC 1771 (Draft Standard), März 1995, http://www.ietf.org/rfc/rfc1771.txt . Obsoleted by RFC 4271. 3

[RFB 01] Ramakrishnan, K., S. Floyd und D. Black: *The Addition of Explicit Congestion Notification (ECN) to IP*. RFC 3168 (Proposed Standard), September 2001, http://www.ietf.org/rfc/rfc3168.txt . Updated by RFCs 4301, 6040. 25, 26

[Roel 05] Roelle, H.: *Eine dienstorientierte Methodik zur Koppelung von Netz–QoS–Architekturen*. Dissertation, Juli 2005. 25

[RSC+ 02] Rosenberg, J., H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley und E. Schooler: *SIP: Session Initiation Protocol*. RFC 3261 (Proposed Standard), Juni 2002, http://www.ietf.org/rfc/rfc3261.txt . Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141. 25

[RTF+ 01] Rosen, E., D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li und A. Conta: *MPLS Label Stack Encoding*. RFC 3032 (Proposed Standard), Januar 2001, http://www.ietf.org/rfc/rfc3032.txt . Updated by RFCs 3443, 4182, 5332, 3270, 5129, 5462, 5586. 25

[RVC 01] Rosen, E., A. Viswanathan und R. Callon: *Multiprotocol Label Switching Architecture*. RFC 3031 (Proposed Standard), Januar 2001, http://www.ietf.org/rfc/rfc3031.txt . Updated by RFC 6178. 1, 25

[SPG 97] Shenker, S., C. Partridge und R. Guerin: *Specification of Guaranteed Quality of Service*. RFC 2212 (Proposed Standard), September 1997, http://www.ietf.org/rfc/rfc2212.txt . 25

[Tane 02] Tanenbaum, Andrew S.: *Computer Networks*. Prentice Hall Professional Technical Reference, 4th Auflage, 2002. 2

[VTRB 97] Vixie, P., S. Thomson, Y. Rekhter und J. Bound: *Dynamic Updates in the Domain Name System (DNS UPDATE)*. RFC 2136 (Proposed Standard), April 1997, http://www.ietf.org/rfc/rfc2136.txt . Updated by RFCs 3007, 4035, 4033, 4034. 87

[Wroc 97a] Wroclawski, J.: *Specification of the Controlled-Load Network Element Service*. RFC 2211 (Proposed Standard), September 1997, http://www.ietf.org/rfc/rfc2211.txt . 25

[Wroc 97b] Wroclawski, J.: *The Use of RSVP with IETF Integrated Services*. RFC 2210 (Proposed Standard), September 1997, http://www.ietf.org/rfc/rfc2210.txt . 25