Vorschlag einer Architektur für Software Defined Networks

Andy Georgi, Jupp Müller, Wolfgang Wünsch, Wolfgang E. Nagel

Zentrum für Informationsdienste und Hochleistungsrechnen
TU Dresden
Zellescher Weg 12-14
01062 Dresden
Andy.Georgi@tu-dresden.de
Jupp.Mueller@zih.tu-dresden.de
Wolfgang.Wuensch@tu-dresden.de
Wolfgang.Nagel@tu-dresden.de

Abstract: Die zunehmende Vernetzung und die Benutzung und Bereitstellung von Cloud-Diensten stellt neue Anforderungen an bestehende Netzwerkarchitekturen. Anstatt wie bisher auf einzelne Probleme mit immer neuen proprietären Lösungen zu reagieren, stellt das Konzept der Software Defined Networks eine offene Architektur zur Verfügung, mit der es möglich ist, verschiedene Probleme wie ineffizientes Routing, ungenutzte Kapazitäten oder manuelle Konfiguration der Netzwerkkomponenten zu lösen. In dieser Arbeit wird eine Architektur vorgestellt, in welcher die beiden offenen Standards OpenFlow und NSI eingesetzt werden, um Anwendungen dynamisch und mittels einer zentralen Sicht auf das Netzwerk Ressourcen zur Verfügung zu stellen.

1 Einführung

Netzwerke sind eine der wenigen verbliebenen Ressourcen, die nach wie vor nicht bedarfsorientiert angeboten werden. Unabhängig von den Spezifikationen der zu sendenden Daten, werden diese in der Regel nach dem Best-Effort-Prinzip übertragen, ohne dass der Nutzer Einfluss darauf nehmen kann. Gleichzeitig nimmt der Datenverkehr, insbesondere durch die Bereitstellung und Nutzung externer Dienste, stetig zu. Dies führt dazu, dass die Kapazitäten in gleichem Maße erhöht werden. Da diese allerdings nicht über den gesamten Zeitraum benötigt werden, stellt diese Vorgehensweise keine effiziente Lösung dar. Ökonomisch sinnvoller ist es hingegen, die verfügbaren Ressourcen besser auszulasten, wobei allerdings kein negativer Einfluss auf die Verbindungsqualität entstehen darf. Lösungsansätze scheitern dabei häufig an der Kompatibilität zwischen den Netzwerkkomponenten verschiedener Hersteller. Mit der Standardisierung und Einführung von OpenFlow [Fou12] steht nun eine Schnittstelle zur Verfügung, welche zum einen die feingranulare Steuerung von Flows ermöglicht und zum anderen Informationen über den aktuellen Status der Ressourcen bereitstellt. Auf Basis dieses Protokolls ist es möglich, eine Netzwerkarchitektur zu entwickeln, mit der eine bedarfsorientierte Datenübertragung angeboten werden kann.

In dieser Arbeit werden ausgehend von einer Datenverkehrsanalyse in Kapitel 2 die Anforderungen an eine effiziente und bedarfsorientierte Netzwerkarchitektur definiert (Kapitel 3). Im Anschluss wird in Kapitel 4 darauf aufbauend eine neue Netzwerkarchitektur vorgeschlagen und der Kommunikationsablauf darin beschrieben. Abschließend erfolgt in Kapitel 5 die Zusammenfassung.

2 Datenverkehr in Rechenzentren

Die Definition einer effizienten Netzwerkarchitektur kann ausschließlich auf Basis der Charakteristika aktueller Datenströme erfolgen. Dazu wird in dieser Arbeit auf bereits erfolgte Datenverkehrsanalysen ([KSG+09] und [BAAZ09]) zurückgegriffen. Darauf aufbauend werden die Anforderungen an die neue Netzwerkarchitektur abgeleitet.

Allgemein wird in Rechenzentren – und oftmals auch darüber hinaus – eine hierarchische Netzwerkstruktur eingesetzt. Dabei unterscheidet man zwischen den Core-Elementen auf der obersten und den Edge-Elementen auf der untersten Ebene. Zum Teil existiert darüber hinaus eine dritte Ebene zwischen den Core- und Edge-Elementen, welche dann als Aggregationsebene bezeichnet wird.

Bei näherer Betrachtung der Datenströme auf den verschiedenen Ebenen kann festgestellt werden, dass lediglich 60% der Core- und Edge-Links aktiv genutzt werden. Darüber hinaus wurde in dem gesamten Messzeitraum von insgesamt 10 Tagen zu keinem Zeitpunkt eine vollständige Link-Auslastung erreicht. Im Detail betrug die Auslastung maximal 45% im Core- und maximal 20% im Edge-Bereich. Dies lässt zunächst darauf schließen, dass mit Overprovisioning die Anforderungen der Anwendungen im Best-Effort Betrieb erfüllt werden können. Dem gegenüber stehen allerdings Datenverlustraten von bis zu 75% innerhalb eines Messintervalls von 5 Minuten im Core- und Aggregation-Bereich. Dies weist auf die Existenz von sog. „Bursty Traffic" hin. Dabei wird in einer kurzen Zeitspanne ein Datenvolumen erzeugt, welches bei der Übertragung die Kapazitäten einer Teilstrecke zwar übersteigt, und somit die Paketverluste verursacht, allerdings nur so kurz auftritt, dass die Statistiken in dem Messintervall nicht wesentlich beeinträchtigt werden. So ist es möglich, dass auf der einen Seite eine Auslastung von maximal 60% erreicht wird, auf der andere Seite allerdings Paketverlustraten von bis zu 75% auftreten.

Auf Basis dieser empirischen Beobachtungen lassen sich bereits erste Anforderungen an eine effiziente Netzwerkarchitektur ableiten. Zur Vermeidung von Paketverlusten ist eine Lastverteilung auf die nicht genutzten bzw. gering ausgelasteten Verbindungen erforderlich. Diese Funktionalität wird zwar bereits von einigen Protokollen, wie bspw. Multipath TCP [RBP+11], unterstützt, setzt aber zumeist eine stabile Kommunikationsmatrix voraus. Sowohl in [KSG+09] als auch in [BAAZ09] wurde gezeigt, dass die Menge der nicht genutzten Verbindungen stetig variiert. Darüber hinaus sind feingranularere Messintervalle erforderlich, als sie von aktuellen SNMP Implementierungen bereitgestellt werden, um auch kurzzeitig auftretende Spitzen verteilen zu können. Die Anforderung, dass durch das Monitoring der Datenverkehr nicht beeinträchtigt wird, bleibt dabei bestehen.

Wie in [KSG+09] gezeigt wird, bestehen über 80% der Verbindungen weniger als 10 Se-

kunden und lediglich 0,1% länger als 200 Sekunden. Somit ist es nicht ausreichend ausschließlich die Zeit für die Erkennung eines Ereignisses zu minimieren, sondern auch die darauf folgende Reaktionszeit, da Unterbrechungen oder Verzögerungen auf kurze Verbindungen einen höheren Einfluss haben als auf Langzeitübertragungen. Dies erfordert demzufolge neben den feingranulareren Messintervallen zusätzlich einen effizienten Routing-Algorithmus, welcher in Abhängigkeit der aktuellen Messdaten und zugrundeliegenden Topologie eine nahezu optimale Entscheidung über die Verteilung der Datenströme trifft, ohne die Verzögerungszeit beim Verbindungsaufbau signifikant zu erhöhen.

3 Anforderungen an moderne Netzwerkarchitekturen

In diesem Abschnitt werden ausgewählte Anforderungen beschrieben, welche heutige und zukünftige Nutzungsmodelle an Netzwerkarchitekturen stellenwas zu dem Konzept der Software Defined Networks (SDN) führt. Dabei soll keinesfalls Vollständigkeit erreicht werden. Stattdessen werden einige Beispiele für die Vorteile aufgezeigt, die eine SDN-Architektur bietet.

Während es durchaus möglich ist, mit konventionellen Mitteln, neuen proprietären Protokollen und spezieller Hardware jedes einzelne Problem für sich zu lösen, so bietet die vorgeschlagene SDN-Architektur einen generellen und flexibleren Ansatz zur Lösung der Probleme.

3.1 Adaptives Routing

Im Vergleich zur Weiterleitung von Paketen mittels statischer Tabellen bietet adaptives Routing sowohl die Möglichkeit von Performanceverbesserungen im Netzwerk, stellt aber auch neue technische Herausforderungen. Pakete im Netzwerk folgen beim adaptiven Routing nicht statischen Pfaden, sondern können, je nach Zustand des Netzwerks auf unterschiedlichen Wegen zu ihrem jeweiligen Ziel weitergeleitet werden. Damit ist es möglich, automatisiert und schnell auf Ausfälle einzelner Netzwerkkomponenten sowie auf Paketstau innerhalb dieser Komponenten zu reagieren. In diesen Fällen können Router im Netzwerk die Pakete auf weniger belasteten Pfaden zum Ziel leiten und dadurch Ausfälle und Überlastungen auf dem ursprünglichen Pfad kompensieren.

Weiterhin ist es möglich, andere Metriken wie Energieeffizienz zu optimieren, indem zum Beispiel bei Situationen mit niedriger Last Teile des Netzwerks abgeschaltet werden [HSM+10].

Durch adaptives Routing können Netzwerke näher an ihrer maximalen Kapazität betrieben werden, ohne in der Qualität der auf dem Netzwerk angebotenen Dienste Nachteile in Kauf nehmen zu müssen. Google ist es zum Beispiel gelungen, die Auslastung des Netzwerks zwischen ihren Datenzentren signifikant zu erhöhen [Hoe12].

3.2 Quality of Service on Demand

Frühere Netzwerkarchitekturen vergaben Quality of Service-Richtlinien nach statischen Parametern wie Quelle, Ziel, Ports oder anderen Parametern. Diese Vergabe wurde durch manuelle Konfiguration von Routern oder Switches durchgeführt und nur selten geändert.

In neuartigen Netzwerkarchitekturen orientiert sich das Netzwerk an den Anforderungen der Anwendungen (Quality of Service on Demand). Wenn eine bestimmte Dienstgüte erforderlich ist, werden automatisiert Pfade im Netzwerk für den benötigten Zeitraum reserviert. Ein manuelles Konfigurieren von Routern, je nach Lage der Anwendung im Netzwerk, entfällt.

3.3 Netzwerkvirtualisierung

Während es bei der Virtualisierung von Computerhardware in den letzten Jahren viele Fortschritte gegeben hat, ist ein ähnlicher Grad von Virtualisierung von Netzwerken noch nicht erreicht worden. Netzwerkvirtualisierung ist kein klar definierter Begriff und kann aus unterschiedlichen Perspektiven betrachtet werden.

Im hier beschriebenen Fall bedeutet dies, dass Anwendungen Netzwerkressourcen zur Verfügung gestellt bekommen, die abstrahiert vom physischen Netzwerk sind. Die Anwendungen greifen wie gewohnt auf Netzwerkressourcen zu. Eine Zwischenschicht stellt die Funktion mittels des physischen Netzwerks sicher. Aus der Virtualisierung ergeben sich in dynamischen Umgebungen einige Vorteile, denn im Gegensatz zu physischen Ressourcen können virtuelle schnell und automatisiert erstellt, gelöscht, migriert, gesichert und pausiert werden. Ein Beispiel dafür ist ein Hardwareausfall, der die Migration der virtuellen Maschine auf einen anderen Server zur Folge hat. Ein virtualisiertes Netzwerk kann darauf in Echtzeit reagieren und die mit der virtuellen Maschine assoziierten Ressourcen wie zum Beispiel IP-Addressen, Verbindungen zu anderen Computern oder reservierte Bandbreite mit umziehen. Da die für die Maschine sichtbaren Ressourcen virtuell sind, ist der Umzug für sie transparent und alle Netzwerkverbindungen bleiben erhalten.

3.4 Dynamische Konfiguration

In klassischen Netzwerken erfolgt die Konfiguration von Komponenten weitestgehend manuell. Der Administrator der Komponente greift auf die Konfiguration zu und nimmt Änderungen vor. Dies führt zu temporär inkonsistenten Zuständen, zum Beispiel in der Zeit zwischen der Konfiguration eines Routers und der Konfiguration seines Gegenstücks.

In Software Defined Networks wird die Konfiguration des Netzwerks von einer zentralen Instanz mit globaler Sicht auf die Topologie verwaltet. Diese höhere Abstraktion erlaubt es, die Konfigurationen der einzelnen Komponenten stets in einem konsistenten Zustand zu halten und den laufenden Produktivbetrieb auch bei umfangreichen Konfigura-

tionsänderungen aufrecht zu erhalten. Dadurch wird die Änderung von Konfigurationen einfacher und schneller. Das Netzwerk kann ständig den Gegebenheiten angepasst werden und wird dynamisch.

4 Architekturvorschlag für Software Defined Networks

Die in Kapitel 3 beschriebenen Anforderungen an eine effiziente und bedarfsorientierte Netzwerkarchitektur sind mit aktuell im Einsatz befindlichen Technologien und dazugehörigen Protokollfamilien nicht realisierbar. Aus diesem Grund schlagen wir in diesem Kapitel eine neue Software Defined Network Architektur vor und beschreiben den Kommunikationsablauf in einer solchen Umgebung.

4.1 Architekturbeschreibung

Unser Vorschlag für eine effiziente und anwendungsorientierte Netzwerkarchitektur besteht aus insgesamt drei Ebenen und ist in Abbildung 1 dargestellt. Auf der untersten Ebene wird die zugrundeliegende Infrastruktur definiert. Dabei handelt es sich um eine heterogene und durch die Hersteller geprägte Umgebung. Der interne Aufbau der Netzwerkelemente, sowie die Verarbeitung der Datenströme können dabei für darüber liegende Schichten verborgen bleiben und weiterhin proprietär gestaltet sein. Auch die Verknüpfung der Elemente unterliegt keinerlei Restriktionen und kann sich dynamisch zur Laufzeit verändern. Entscheidend ist die Auslagerung der Kontrollfunktion in die darüberliegende Ebene und deren Anbindung über eine standardisierte Schnittstelle.

Um Interoperabilität in einer heterogenen Umgebung zu gewährleisten, ist eine standardisierte Schnittstelle erforderlich. Wir schlagen zu diesem Zweck die Verwendung von OpenFlow vor. Entwickelt wurde dieses Protokoll, um es Forschern zu ermöglichen, neue Netzwerkprotokolle in einer Produktivumgebung zu testen, ohne dabei den Produktivbetrieb zu beeinflussen [MAB+08]. Dabei wird das erste Paket eines Flows an einen externen Controller gesendet, der entscheidet an welchen Ausgangsport dieses und alle nachfolgenden Pakete des Flows weitergeleitet werden. Die Identifikation kann dabei anhand frei definierbarer Bytes innerhalb des Frames erfolgen, wodurch dieses Verfahren unabhängig vom eingesetzten Netzwerkprotokoll ist. Der Controller erstellt zudem den entsprechenden Eintrag in der Forwarding Table der Netzwerkkomponente, sodass nachfolgende Pakete die Geschwindigkeit der Backplane nutzen können.

Die Standardisierung von OpenFlow erfolgt seit 2011 durch die Open Network Foundation (ONF) [Fou13]. Zum Zeitpunkt der Entstehung der vorliegenden Arbeit ist OpenFlow in der Version 1.3.1 spezifiziert [Fou12] und damit im Vergleich zu anderen Ansätzen, wie bspw. dem Software Driven Network Protokoll der IETF [Gro11], am weitesten fortgeschritten. Zudem sind zahlreiche Netzwerkgerätehersteller aktives Mitglied der ONF und unterstützen OpenFlow bereits bis Version 1.2 in ihren Komponenten. Darüber hinaus wurden zu dem ursprünglichen Port-Forwarding weitere Funktionalitäten, wie das Sammeln

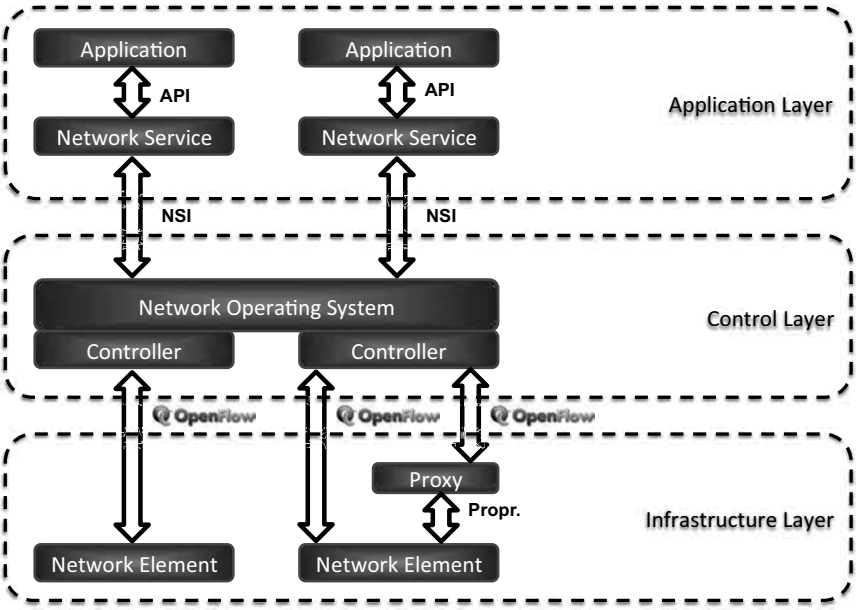

Abbildung 1: Schichtenmodell der vorgeschlagenen Architektur

von Statistiken oder die Spezifizierung von QoS-Parametern für einen Flow, aufgenommen. Dies ermöglicht die Reaktion auf Ereignisse zur Laufzeit und die Bereitstellung von Netzwerkressourcen, welche sich an den Bedürfnissen des Nutzers orientieren.

Da die Umstellung auf OpenFlow kompatible Geräte schrittweise erfolgt, empfehlen wir in der Übergangsphase den Einsatz einer Proxy-Lösung. Dabei nimmt eine Middleware die OpenFlow Nachrichten entgegen und übersetzt diese in proprietäre Systemnachrichten die von der Netzwerkkomponente interpretiert werden können. Die Implementierung wird voraussichtlich eng an die Management-Software des Netzwerkelements gekoppelt sein, welche bereits von zahlreichen Herstellern zu ihren Geräten angeboten wird. Damit ist es bereits möglich Statistiken mehrerer Geräte zu aggregieren und diese von einem zentralen Punkt aus zu steuern. Mit dieser Verfahrensweise kann in der vorgestellten Architektur Kompatibilität zu bestehenden Infrastrukturen hergestellt werden. Allerdings entsteht durch das Einfügen einer Middleware zusätzlicher Overhead, welcher mit OpenFlow kompatiblen Netzwerkelementen vermieden werden kann.

Die Kontrollebene implementiert die Funktionalitäten, welche aktuell noch von dem Netzwerkelement selbst bereitgestellt werden und bildet die Schnittstelle zur Anwenderebene. Die Trennung von Control- und Data-Plane ermöglicht eine freie und herstellerübergreifende Konfiguration und Steuerung von Netzwerkressourcen. Darüber hinaus können Entscheidungen auf der Basis einer globalen Sicht auf die darunter liegende Infrastruktur getroffen werden, was zum einen zu effizienteren Entscheidungen führen kann, zum anderen aber auch höheren Berechnungsaufwand verursacht. An der Stelle ist darauf zu achten,

dass keine negativen Auswirkungen durch erhöhte Verzögerungszeiten beim Verbindungsaufbau entstehen.

Die Bestandteile der Kontrollebene sind ein oder mehrere Controller auf der Seite der Infrastrukturebene, sowie auf Anwendungsseite ein Network Operating System (NOS). Die Aufgabe des Controllers ist die Steuerung der Netzwerkelemente. Dazu gehört bspw. das Eintragen eines Flows in die Forwarding Table oder auch die Konfiguration von QoS-Parametern für eine Verbindung. Weiterhin liefert er Informationen über die Infrastrukturebene an das NOS. Dieses nimmt Verbindungsanfragen aus der Anwendungsebene entgegen und berechnet die dafür notwendigen Netzwerkressourcen. Dabei können zusätzliche Rahmenbedingungen, wie bspw. Kosteneffizienz, Übertragungszeit oder Datenrate spezifiziert werden. Im Anschluss an die Berechnung der Kommunikationsstrecke kann diese bei den verschiedenen Controllern auf dem Weg zum Empfänger registriert werden, sodass Pakete beim Eintreffen ohne weitere Interaktion mit dem NOS weitergeleitet werden können. Zudem stellt das NOS den sicheren Zugriff auf die Netzwerkressourcen im Multiuser-Betrieb und die Interoperabilität zwischen mehreren Controllern sicher.

Zur Kommunikation zwischen Anwendungs- und Steuerungsebene schlagen wir den Einsatz des Network Service Interfaces (NSI) [Rad12] vor, welches durch Vertreter von DANTE, i2CAT, ESNet et al. spezifiziert wird. NSI bildet die Schnittstelle zwischen einem Service Requestor und einem Service Provider zur Anfrage einer Transportverbindung. Zu den bereitgestellten Funktionalitäten gehören bspw. Scheduling, Reservierung von Ressourcen, Monitoring oder die Abfrage von Netzwerkkapazitäten sowie Topologieinformationen. Somit ermöglicht es NSI, dass die auf Infrastrukur- und Kontrollebene bereitgestellten Funktionalitäten zusammengefasst und an die Anwendungsebene weitergereicht werden können. Zudem trägt die Schnittstelle durch Abstraktion zur besseren Nutzbarkeit bei.

Aus der Anwendungsebene heraus spezifiziert der Nutzer über seine Applikation - falls diese NSI unterstützt - den Bedarf an Ressourcen und teilt diesen direkt dem NOS mit, bzw. schickt die Anfrage über einen Network Service. Dies können Softwarewerkzeuge sein, wie bspw. GLOBUS, IRODS oder gLite welche NSI implementieren und die Anfragen an die Kontrollebene weiterleiten. Als Antwort erhält der Nutzer die Information, ob und zu welchem Zeitpunkt die Verbindung zu den spezifizierten Bedingungen möglich ist. Vorgesehen ist nach wie vor die Möglichkeit zur Übertragung der Daten im Best-Effort-Betrieb, sodass grundsätzlich immer eine Kommunikation stattfinden kann.

4.2 Beschreibung des Kommunikationsverlaufs

Die Spezifikation von Randbedingungen für eine Verbindung führt dazu, dass Datenübertragungen nicht ausführbar sind, falls die Anforderungen die noch verfügbaren Ressourcen übersteigen. Insbesondere wenn eine Terminierungszeit gefordert ist, kann dies dazu führen, dass eine Anfrage zurückgewiesen wird. Um dennoch eine Datenübertragung unabhängig von der Auslastung zu gewährleisten, empfehlen wir nach wie vor, einen Teil der Kapazitäten im Best-Effort-Betrieb bereitzustellen, wobei Nachrichten wie bisher unter

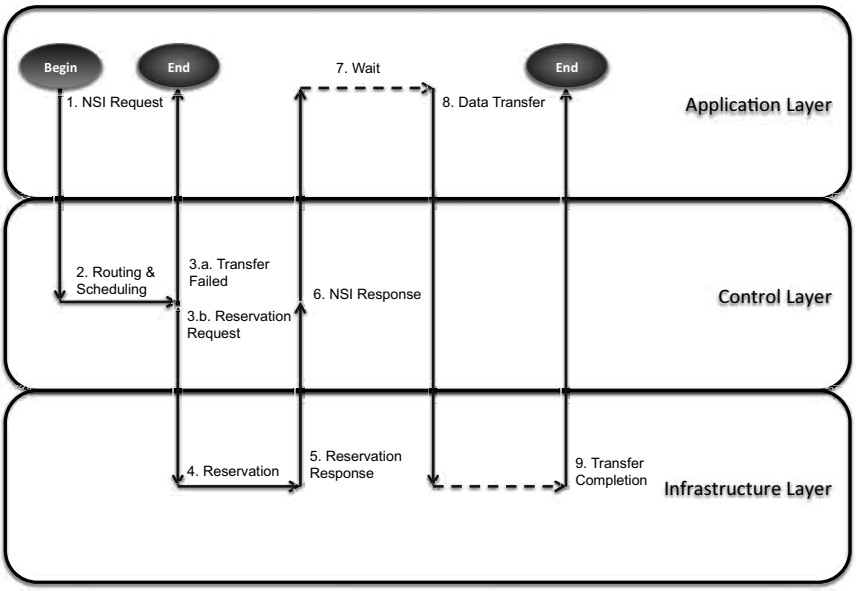

Abbildung 2: Kommunikationsablauf

Verwendung der Internetprotokollfamilie versendet und empfangen werden. Da der Kommunikationsablauf hierbei hinreichend bekannt ist, beschränken wir uns an dieser Stelle auf die Beschreibung einer bedarfsorientierten Datenübertragung.

In Abbildung 2 ist der zeitliche Ablauf einer bedarfsorientierten Kommunikation über die in Abschnitt 4.1 definierten Ebenen dargestellt. Dabei wird zunächst der Bedarf in der Anwendung formuliert und entweder direkt oder über einen Dienst mit Hilfe des Network Service Interface (NSI) an die Kontrollebene übertragen. Zu den Anforderungen können wie in Kapitel 3 beschrieben Start- und Endzeitpunkt einer Kommunikation, Kriterien für die Wegfindung oder auch QoS-Parameter gehören.

Der NSI Request wird auf der Kontrollebene von dem Network Operating System (NOS) entgegengenommen und ausgewertet. Dabei wird geprüft, ob der Nutzer die Berechtigung besitzt, die Ressourcen anzufordern, und ob diese auf Infrastrukturebene verfügbar sind. Ist dies nicht der Fall, so wird dem Nutzer mitgeteilt, dass eine Verbindung unter diesen Rahmenbedingungen nicht durchführbar ist. Daraufhin sind entweder die Anforderungen anzupassen, oder die Übertragung muss im Best-Effort-Betrieb erfolgen. Besitzt der Nutzer hingegen die notwendigen Berechtigungen und sind die Ressourcen verfügbar, erfolgt die Berechnung des Weges durch das Verbindungsnetzwerk auf Basis der Nutzervorgaben und der aktuell vorliegenden Statusinformationen. Diese werden unter Verwendung des OpenFlow Protokolls in regelmäßigen Abständen oder mit jeder Kommunikationsanfrage von der Infrastrukturebene abgefragt. Letzteres kann allerdings bei hohem Verkehrsaufkommen zu einer Überlastung und damit zu erhöhten Verzögerungszeiten führen, weshalb eine regelmäßige Aktualisierung vorzuziehen ist.

Ist der Weg zwischen Sender und Empfänger entsprechend der Rahmenbedingungen definiert, kann die Reservierung durch die Controller erfolgen. Dazu wird die Verbindung zusammen mit den ggf. vorgegebenen QoS-Parametern unter Verwendung von OpenFlow bei den einzelnen Netzwerkkomponenten registriert und die Ausgangsports in die Forwarding Tables eingetragen. Alle Einträge sind mit einer Lebenszeit versehen und werden nach Ablauf automatisch aus der Forwarding Table entfernt. Ein vorzeitiges Entfernen oder eine Verlängerung der Gültigkeit eines Eintrages kann lediglich auf Anweisung des NOS erfolgen.

Ist die Reservierung auf Infrastrukturebene abgeschlossen, wird der Anwendung über das NOS der Zeitraum mitgeteilt, in dem die Reservierung gültig ist. Da es dem Nutzer freisteht, diese in Anspruch zu nehmen, sollten die reservierten Kapazitäten bis zur Nutzung durch die bedarfsorientierte Übertragung dem Best-Effort-Betrieb bereitgestellt werden.

5 Zusammenfassung

In dieser Arbeit wurde eine Architektur für Software Defined Networks vorgestellt, die mit einer abstrakten Sicht auf die physischen Netzwerkkomponenten eine Möglichkeit bietet, um auf verschiedene Anforderungen zu reagieren. Das Ziel ist hierbei nicht, einzelne Probleme mit minimalem Aufwand zu lösen, sondern eine stabile Plattform zu etablieren, mit der sich eine Vielzahl von Anforderungen an moderne Netzwerke auf eine generische Art und Weise lösen lassen.

Der Fokus liegt dabei auf der Sicht der Anwendung, die Anforderungen an das Netzwerk stellt. Diese Anforderungen werden an Network Services formuliert, von diesen formalisiert und an eine zentrale Steuerungsinstanz, das Network Operating System, weitergegeben. Dieses wiederum nimmt mit der Hilfe von Controllern die Konfiguration der einzelnen Netzwerkkomponenten vor, um den gestellten Anforderungen zu entsprechen.

Wir haben zwei Standards beschrieben, die zwischen der Anwendungsschicht und der Kontrollschicht auf der einen, der Kontrollschicht und der Infrastrukturschicht auf der anderen Seite, für Interoperabilität sorgen. Network Services benutzen den NSI-Standard, um ihre Anforderungen an das Netzwerk zu formulieren und eine Antwort auf ihre Anfragen zu erhalten. Die Controller verwenden OpenFlow, um die Konfiguration der physischen Netzwerkkomponenten vorzunehmen.

Der wesentliche Unterschied und der große Vorteil gegenüber klassischen Netzwerkarchitekturen liegt in der globalen Sicht des Network Operating System auf das Netzwerk. Während anderen Lösungen für Probleme wie ineffizientes Routing oder Quality of Service nur ihre lokale Sicht auf das Netzwerk zur Verfügung steht, sind Software Defined Networks in der Lage, ganzheitliche und damit bessere Lösungen zu finden und gleichzeitig dynamisch auf Veränderungen zu reagieren.

Der zweite wichtige Vorteil ist die abstrakte Sicht auf das Netzwerk, welche das Network Operating System den Anwendungen zur Verfügung stellt. Anwendungen müssen nicht mehr selbst die Eigenschaften des Netzwerks vermessen, sondern können ihren Bedarf mittels definierter Schnittstellen anmelden und bekommen eine verbindliche Antwort, ob

ihren Anforderungen entsprochen werden kann. Das Netzwerk wird somit zu einem frei konfigurierbarem Dienst.

Literatur

[BAAZ09] Theophilus Benson, Ashok Anand, Aditya Akella und Ming Zhang. Understanding data center traffic characteristics. In *Proceedings of the 1st ACM workshop on Research on enterprise networking*, WREN '09, Seiten 65–72, New York, NY, USA, 2009. ACM.

[Fou12] Open Network Foundation. OpenFlow Switch Specification, Version 1.3.1 (Wire Protocol 0x04). `https://www.opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.3.1.pdf`, September 2012.

[Fou13] Open Network Foundation. Webseite. `https://www.opennetworking.org/index.php`, 2013. zuletzt abgerufen am 03.01.2013.

[Gro11] IETF Network Working Group. Software Driven Networks: Use Cases and Framework. `http://tools.ietf.org/html/draft-stiliadis-sdnp-framework-use-cases-01`, Oktober 2011.

[Hoe12] Urs Hoelzle. OpenFlow @ Google. Open Networking Summit, 2012.

[HSM+10] Brandon Heller, Srini Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee und Nick McKeown. ElasticTree: saving energy in data center networks. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10, Seiten 17–17, Berkeley, CA, USA, 2010. USENIX Association.

[KSG+09] Srikanth Kandula, Sudipta Sengupta, Albert Greenberg, Parveen Patel und Ronnie Chaiken. The nature of data center traffic: measurements & analysis. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, Seiten 202–208, New York, NY, USA, 2009. ACM.

[MAB+08] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker und Jonathan Turner. OpenFlow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, Marz 2008.

[Rad12] Radek Krzywania (PSNC), Joan A. Garcia -Espin (i2CAT), Chin Guok, Inder Monga (ESnet), Jeroen van der Ham (UvA), Tomohiro Kudoh (AIST), John MacAuley (SURFnet), Joe Mambretti (Northwestern University), Guy Roberts (DANTE), Jerry Sobieski (NORDUnet), Alexander Willner (University of Bonn). Network Service Interface - gateway for future network services. 2012.

[RBP+11] Costin Raiciu, Sebastien Barre, Christopher Pluntke, Adam Greenhalgh, Damon Wischik und Mark Handley. Improving datacenter performance and robustness with multipath TCP. In *Proceedings of the ACM SIGCOMM 2011 conference*, SIGCOMM '11, Seiten 266–277, New York, NY, USA, 2011. ACM.