OpenFlow

Sebastian Rampfl
Betreuer: Dipl.-Ing. Dirk Haage
Seminar Innovative Internet Technologien und Mobilkommunikation SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: rampfl@in.tum.de

ABSTRACT

Das Internet befindet sich im ständigen Wandel. Neue Angebote entstehen, andere verschwinden wieder. Auch neue Technologien und Netzwerk-Protokolle werden entwickelt. Doch diese zu etablieren und in das Internet zu integrieren erweist sich als schwierig, nicht zuletzt deswegen, weil es den Forschern an Möglichkeiten fehlt, ihre Ideen und Technologien unter realtitätsnahen Bedingungen zu testen. OpenFlow ist ein neues Protokoll, welches genau das bietet. Es gibt Forschern Kontrolle über bestehende Netz-Infrastruktur und ermöglicht ihnen somit ihre Ideen in größeren Netzen zu testen, und das ohne, dass der normale Traffic der Infrastruktur, also der Datenverkehr, der im normalen Betrieb entsteht, davon beeinflusst wird. Doch nicht nur für die Forschung ist OpenFlow ein vielversprechendes Protokoll, denn diese neue Möglichkeit der Einflussnahme auf ein Netzwerk eröffnet eine ganze Reihe weiterer Möglichkeiten für technische Innovationen.

Keywords

OpenFlow, Virtualisierung, FlowVisor

1. EINLEITUNG

Technischer Fortschritt und Innovationen auf dem Gebiert der Netzwerktechnik und der Netzdienste sind wichtige Faktoren für den Ausbau und die Optimierung von Netzwerken und nicht zuletzt des Internets. Wer sich mit Netzen und den verwendeten Techniken und Protokollen auseinandersetzt stellt fest, dass sich neue experimentelle Protokolle meist nur schwer in bestehende Netzinfrastruktur integrieren lassen. Man spricht hier von der „Verknöcherung“ von Netzwerken oder allgemein des Internets. Das hat zum einen den Grund, dass sich Protokolle nicht einfach durch neue ersetzen lassen. So kann beispielsweise in einem Netzwerk nicht auf einem Teil der Infrastruktur das IP-Protokoll laufen während auf anderen Geräten innerhalb des Netzes ein alternatives IP-Protokoll zum Einsatz gebracht wird. Die in Netzwerken eingesetzten Geräte werden von den Herstellern zudem größtenteils als geschlossene, nicht programmierbare Systeme konzipiert, die kaum Einflussnahme auf die darauf laufenden Protokolle ermöglichen. Gründe dafür sind zum einem der Wunsch der Hersteller die Architektur und Funktionsweise ihrer Hardware zu verschleiern um ihren Wettbewerbern nicht unnötig Informationen über ihre Technologien zu offenbaren, zum anderen besteht die berechtigte Befürchtung das Veränderungen an ihren Geräten durch Forscher die Stabilität ganzer Netzwerke gefährdet seien könnte. Alle diese Umstände führen zu der Erkenntnis, dass das Implementieren und Testen experimenteller Protokolle in bestehender Netz-Infrastruktur wie zum Beispiel einem Campus-Netzwerk kaum möglich ist. Alternativ gäbe es auch die Möglichkeit ein Netz aus programmierbaren Switches und Routern aufzubauen, um auf dieser Hardware die neuen Protokolle laufen zu lassen. Dies hätte den Vorteil, dass diese Variante, abgeschottet von anderen Netzen, wie einem Campus-Netzwerk, normalen Traffic nicht negativ beeinflussen kann. Ein solches reines Forschungs-Netz kommt für die meisten Forscher und Forschungseinrichtungen allerdings nicht in Frage, da es mit zu hohen Kosten verbunden ist wenn das Netzwerk eine Größe und Portdichte erreichen soll, die ein Testen unter realitätsnahen Bedingungen ermöglicht. Genau hier setzt OpenFlow an.

In diesem Paper werden das OpenFlow-Protokoll, seine Funktionsweise und die Möglichkeiten, die es für die Forschung und weitere Anwendungen bietet, beschrieben. Das Paper ist dabei wie folgt gegliedert. In Kapitel 2 werden zuerst die Anforderungen erläutert, die sich aus den in Kapitel 1 geschilderten Problemen ableiten. Im Anschluss wird auf die technische Umsetzung von OpenFlow und die Komponenten dieses Systems eingegangen. Dabei wird auch erklärt wie und in welchem Maß die Implementierung von OpenFlow die gestellten Anforderungen erfüllt. Zusätzlich wird noch eine Erweiterung zu OpenFlow erläutert, der so genannte FlowVisor-Controller. In Kapitel 3 beschreibt das Paper dann OpenFlow in der Praxis. Dazu werden zuerst verschiedene Forschungsansätze diskutiert und im Anschluss weitere Beispiele für Arten der Anwendung genannt. Zusätzlich werden in Kapitel 3 noch drei in ihrer Entwicklung weit fortgeschrittene und auf OpenFlow basierende Projekte beschrieben, um noch einmal die umfangreichen Möglichkeiten, die OpenFlow bietet, zu verdeutlichen. In Kapitel 4 werden dann noch allgemeine Fakten zum OpenFlow-Projekt genannt und in Kapitel 5 ein Fazit gezogen.

2. Anforderungen und technische Umsetzung

2.1 Anforderungen an OpenFlow

Die erste Anforderung an das OpenFlow-Protokoll war es, ein Einbinden von Standard-Netz-Hardware zu ermöglichen. Es sollten für den Betrieb also keine speziellen Geräte wie teure offene Systeme notwendig sein. Dadurch wird sichergestellt, dass OpenFlow in normaler Netzwerk-Infrastruktur wie einem Universitäts-Netzes zum Einsatz kommen kann. Gleichzeitig sollte es aber den in einem solchen Netzwerk anfallenden Traffic nicht beeinflussen, d.h. Tests von experimentellen Protokollen dürfen die Stabilität und Zuverlässigkeit des Netzes nicht gefährden. Zusätzlich sollten in einem OpenFlow-Netz Tests zu einer möglichst breiten Palette an Forschungsansätzen realisierbar

sein. Auch auf die Hersteller der Hardware musste dabei Rücksicht genommen werden. Sie dürfen dabei ihre internen Switch- und Router-Implementierungen nicht offenlegen müssen. Im Folgenden wird darauf eingegangen wie diese Anforderungen umgesetzt wurden. Die in Kapitel 1 beschriebenen Probleme und die daraus resultierenden Anforderungen werden in Referenz [1] erläutert.

2.2 Das OpenFlow-Konzept und die technische Umsetzung

Die Idee hinter OpenFlow ist folgende: Über einen Rechner im Netzwerk, dem sogenannten Controller, werden die FlowTables aller OpenFlow-fähigen Switches, Router und Access Points in einem Netzwerk kontrolliert. Das OpenFlow-Protokoll schafft für den Controller eine Schnittstelle in allen Geräten der Netz-Infrastruktur um Flows, also Paketströme, durch das Netzwerk zu lenken. Wie in Abbildung 1 zu sehen, erfolgt das Editieren der FlowTable über die eigens dafür geschaffene Schnittstelle SecureChannel. Diese Schnittstelle muss allerdings schon vom Hersteller der Netzwerk-Geräte ab Werk integriert sein. Damit ist die Anforderung nach der Einsetzbarkeit in einer normalen Netzwerk-Infrastruktur nur bedingt erfüllt, da die Geräte erst ausgetauscht werden müssen, um OpenFlow innerhalb eines bestehenden Netzwerks einsetzen zu können. In einigen Fällen ist es auch möglich durch ein Firmware-Update, also das Aktualisieren der auf der Hardware installierten Betriebssysteme, das OpenFlow-Protokoll in ein Gerät zu integrieren. Im Folgenden wird auf die einzelnen Bestandteile und die genauere Funktionsweise von OpenFlow, wie sie in dem offiziellen Whitepaper [1] und in der OpenFlow-Spezifikation [2] beschrieben ist, eingegangen.

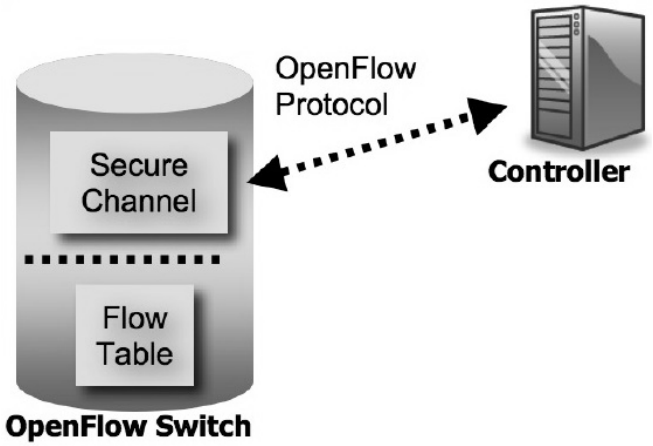

Abbildung 1. FlowTable, SecureChannel, Controller [2]

2.2.1 Flows und FlowTable

Ein Flow ist ein Paketstrom. Dabei wird der Flow durch die Daten im Header der Pakete definiert. Beispielsweise können alle Pakete mit derselben IP-Zieladresse einem Flow zugeordnet werden. Man kann einen Flow aber auch anhand von Ports oder MAC-Adressen festlegen. Grundsätzlich kann jede Information im Header als Regel zur Definition eines Flows dienen. Die FlowTable listet die Flows auf und legt für jeden Flow eine oder mehrere Regeln fest wie das Switch oder der Router mit dem Flow verfahren soll. Wie in Abbildung 2 zu sehen besteht eine FlowTable aus drei Spalten. Header Fields gibt an welche Pakete zu diesem Flow gehören. Im Feld Counters werden Statistiken über den Flow geführt und im Feld Actions finden sich die Befehle die auf die Pakete dieses Flow angewendet werden sollen. So kann das Paket beispielsweise an einen bestimmten Port dirigiert werden, es kann aber auch verworfen werden. Die OpenFlow-Spezifikation legt hier eine Reihe von Befehlen fest die in jedem OpenFlow-fähigen Gerät implementiert seien müssen.

Header Fields	Counters	Actions

Abbildung 2. Die Spalten der FlowTable

2.2.2 OpenFlow-Switch

Die Spezifikation unterscheidet hier zwei unterschiedliche Typen von Switches. Das erste ist das dedicated OpenFlow-Switch, auch OF-only Switch genannt. Es kennt nur die drei wichtigsten Befehle. Der erste Befehl leitet ein Paket zu einem Switch-Port, d.h. zu einer Datenleitung die aus dem Switch zu einem anderen Gerät führt, weiter. Der zweite Befehl sendet das Paket zum Controller und der Dritte verwirft das Paket. Die zweite Variante ist das OF-enabled Switch. Es verfügt über denselben Befehlssatz und zusätzlich noch über die Möglichkeit Pakete an die normale „Processing pipeline“ des Gerätes weiterzuleiten, d.h. die Routing-Entscheidung wird vom Switch selbst ohne Einfluss des OpenFlow-Controllers getroffen. Diese Funktion ist notwendig um normalen Traffic von Forschungs-Traffic zu trennen. Letzterer wird vom Forscher über die FlowTable kontrolliert während der normale Traffic unbeeinflusst davon durch die Processing Pipeline des Switches oder Router bearbeitet und weitergeleitet wird. Der Netzwerk-Administrator legt hierbei durch VLANs fest welche Pakete welchen Traffic zuzuordnen sind und der Forscher hat somit keinen Einfluss auf den normalen Traffic, da dieser nicht in der FlowTable erfasst werden kann. Damit wird schon eine zentrale Anforderung an OpenFlow, die strikte Trennung von Forschungs-Traffic und dem durch den normalen Gebrauch der Netz-Infrastruktur entstehenden Traffic, erfüllt. Die Spezifikation bietet noch eine Reihe weiterer optionaler Befehle an, die nach Ermessen der Hersteller implementiert werden können. So lässt sich beispielsweise auch die VLAN-ID eines Flows modifizieren.

2.2.3 SecureChannel und Controller

Der SecureChannel ist die Schnittstelle zwischen Switch und Controller. Diese Schnittstelle muss im Switch implementiert sein und kann immer nur mit einem Controller kommunizieren. Allgemein kann es in einem OpenFlow-Netzwerk nur einen Controller geben. Der Controller erstellt, editiert und löscht FlowTable-Einträge, d.h. er definiert Flows und legt fest wie mit ihnen verfahren werden soll. Der Forscher implementiert sein experimentelles Protokoll auf dem Controller und dieser steuert den Forschungs-Traffic über die Switches und Router im Netzwerk dann nach den Vorgaben des experimentellen Protokolls. Im Detail passiert dabei Folgendes. Der Forscher sendet von einem Host im Netzwerk Pakete. Das erste OF-Switch, das diese Pakete erhält leitet sie weiter zum Controller, da es die Pakete noch keinen Flow zuordnen kann. Der Controller analysiert den Header der Pakete, definiert diese dann als Flow und schafft entsprechende Einträge in den FlowTables.

Gleichzeitig fügt er diesen Einträgen noch die Befehle hinzu, welche auf den Flow angewandt werden sollen. Nun bewegen sich die Pakete des Forschers nach den von seinem experimentellen Protokoll definierten Regeln durch das Netzwerk. Der Controller kann die von der FlowTable geführten Statistiken zu den Flows abrufen und so die Leistung des getesteten Protokolls analysieren. Hier wird schon ein wichtiger Vorteil von OpenFlow im Vergleich zu einem wie in Kapitel 1 beschrieben programmierbarem Netzwerk deutlich. In Letzterem muss das experimentelle Protokoll auf jeden Switch und Router implementiert werden. Die Implementierung muss dabei den Hersteller-spezifischen Eigenheiten der einzelnen Geräte angepasst werden. In einem OpenFlow-Netzwerk wird das experimentelle Protokoll nur auf dem Controller implementiert, welcher dann durch die standardisierte Schnittstelle in den Geräten die entsprechenden Änderungen durchführt. Die Hersteller müssen dazu die internen Implementierungen ihrer Software und den genauen Aufbau ihrer Hardware nicht offenlegen. Somit wird eine weitere Anforderung an OpenFlow erfüllt. Für den Einsatz von OpenFlow in beispielweise einen Campus-Netzwerk ist es nicht notwendig, dass in jedem Switch, Router oder Access Point das OpenFlow-Protokoll integriert ist. Der Controller hat dann aber nur Kontrolle über die OpenFlow-Geräte und der Forschungs-Traffic kann nur auf diesen Geräten nach den vom dem Controller definierten Regeln beeinflusst werden. Umso größer der Anteil an OpenFlow-Geräten in einem Netzwerk umso komplexere und realitätsnähere Versuche lassen sich damit umsetzen. Die sichere Kommunikation zwischen dem Controller und den SecureChannels der Geräte wird durch den Austausch von Zertifikaten zur Authentifizierung und dem Einsatz von TLS zu Verschlüsselung gewährleistet. Dies ist notwendig um zu verhindern, dass sich ein Teilnehmer im Netzwerk unrechtmäßig Kontrolle über die Switches und Router verschafft. Neben der Sicherheit stellt sich auch die Frage ob ein einzelner Controller überhaupt alle Flows in akzeptabler Geschwindigkeit bearbeiten kann. Die OpenFlow-Autoren kommen aufgrund von Tests zu dem Schluss, dass dies der Fall ist, denn ein gängiger PC kann schon mit bis zu 10.000 Flows in der Sekunde umgehen. Der zentralisierte Controller schafft eine einheitliche Sicht auf das Netzwerk, seine Topologie, seine Auslastung und Veränderungen im Netz und eröffnet dadurch neue Möglichkeiten die vorher nur schwer oder überhaupt nicht zu realisieren waren.

2.2.4 FlowVisor

OpenFlow ist so konzipiert, dass nur ein Controller das OpenFlow-Netz steuert, d.h. es kann immer nur ein experimentelles Protokoll getestet werden. Diesem Problem widmet sich Referenz [3] und liefert als Lösung den FlowVisor-Controller. FlowVisor ist eine Software für den Controller, die es Forschern ermöglicht gleichzeitig mehrere Controller zu betreiben um unterschiedliche Protokolle gleichzeitig im Netzwerk zu testen. Wie in Abbildung 3 zu sehen fungiert FlowVisor hier als transparenter Proxy für weitere Controller. Dazu werden so genannte Slices eingerichtet. Sie werden ähnlich wie Flows definiert, d.h. Pakete werden einem Slice aufgrund ihrer Header-Informationen zugeordnet. Das können beispielsweise alle Pakete mit derselben MAC-Ziel-Adresse, desselben TCP-Quell-Ports oder einer Kombination daraus sein. Die Slices müssen klar voneinander abgetrennt sein und dürfen sich nicht überschneiden. Zusammenfassend kann man sagen, dass Slices aus einem oder mehreren Flows bestehen auf denen exklusiv ein Experiment durchgeführt wird. Wichtig bleibt dabei zu erwähnen, dass FlowVisor diese Slices definiert und dafür sorgt, dass das ein Paket nicht zwei unterschiedlichen Slices zugeordnet werden kann. Jedem mit dem FlowVisor-Controller verbundenen Controller werden ein oder mehrere Slices zugeteilt, die er dann für seine Experimente nutzen kann. Diese können wiederum auch FlowVisor-Controller sein, die die Slices nochmal in weitere Slices aufteilen und weiteren Controllern zuteilen. Dieses Konzept nennt sich Rekursive Delegation. FlowVisor überprüft jede über ihn verschickte OpenFlow-Steuerungs-Nachricht und kann damit auftretende Probleme erkennen und ggf. beheben. Probleme können beispielsweise entstehen wenn ein Controller Flows definiert, die nicht mehr in dem ihm zugeteilten Slice liegen, d.h. er definiert Regeln für Pakete, die ggf. zum Forschungs-Traffic eines anderen Controllers gehören und sich damit beide Controller gegenseitig ihre Protokoll-Tests verfälschen. Der FlowVisor-Controller kann dadurch auch gewisse Verhaltensregeln im Umgang mit dem Forschungs-Traffic und den FlowTables durchsetzen. Die Funktionalität die FlowVisor bietet hätte auch in OpenFlow integriert werden können. Die OpenFlow-Autoren haben das aber bewusst unterlassen, denn dies hätte genau wieder zu der in Kapitel 1 beschriebenen Verknöcherung des Netzes geführt, die die Entwicklung von OpenFlow erst notwendig gemacht hat. Sie streben eine Modularisierung ihres Systems an, in der einzelne Komponenten wie FlowVisor problemlos ausgetauscht werden können.

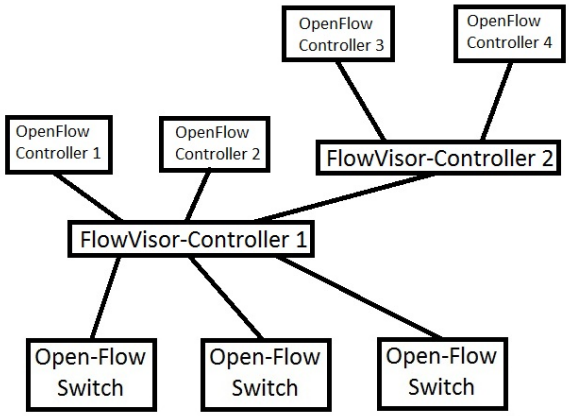

Abbildung 3. Beispiel eines Controller-Verbundes mittels FlowVisor

3. OpenFlow in der Praxis

3.1 Forschungsansätze

OpenFlow ermöglicht eine Vielzahl an Forschungsansätzen. So ließe sich zum Beispiel eine Alternative zum gängigen Routing-Verfahren testen. Wie oben beschrieben würden die neuen Routing-Regeln im Controller implementiert. Die Pakete würden von einem Computer des Forschers in das Netz geschickt, wo der Controller die Pakete erst nach den Regeln des experimentellen Routing-Protokolls als einen oder mehrere Flows definiert und dann alle Switches und Router innerhalb des Netzwerks entsprechend konfiguriert. Der Forscher kann dadurch sein Routing-Protokoll in einem großen Netzwerk testen und sein

Verhalten analysieren. Da man Flows beliebig deklarieren kann ließe sich auch eine Alternative zum IP-Protokoll realisieren. Dazu könnte ein Forschungs-Protokoll beispielsweise nur die MAC-Adressen aller Forschungsrechner im Netz berücksichtigen und eine eigenes System der Adressen-Vergabe auf der Vermittlungsschicht umsetzen. In der Praxis würden die Flows in diesem NON-IP-Netzwerk nur unter der Berücksichtigung der Ethernet-Header eines jeden Paketes definiert.

3.2 Anwendungsbeispiele

Neben der Forschung eröffnet OpenFlow auch noch ein breites Spektrum an zusätzlichen Anwendungsmöglichkeiten wie sie im Whitepaper zu OpenFlow [1] beschrieben werden. So könnte man beispielsweise den gesamten Traffic eines Netzwerks von dem Controller überwachen lassen, um Policies im Netzwerk durchzusetzen. Ein Controller könnte dabei alle VoIP-Pakete verwerfen lassen um diesen Service im OpenFlow-Netz zu sperren. Hierzu wäre dann auch nur ein Eintrag in den FlowTables aller Switches, Router und Access Points notwendig. Denkbar wäre auch ein Aufbau von VLANs mit Hilfe des OF-Protokolls. Die Switches würden dabei die von bestimmten Hosts versendeten Pakete mit VLAN-IDs versehen. So genannte Mobile wireless VoIP-Verbindungen ließen sich auch relativ einfach realisieren. Wenn ein Teilnehmer eines VoIP-Gesprächs den Access Point wechselt könnte der Datenstrom durch den Controller von alten zum neuen Access Point umgeleitet werden ohne, dass ein erneuter Verbindungsaufbau nötig wird. Somit wären die Teilnehmer in der Lage ihr Gespräch ohne Unterbrechung weiterführen. Ein weiteres im Whitepaper aufgeführtes Beispiel ist das Umleiten des Traffic durch ein offenes programmierbares System, wie etwa einen NetFPGA. Dort würde dann jedes Paket überprüft und ggf. modifiziert werden. Dadurch könnten dann verschiedenste Operationen, wie die Suche nach nicht autorisierten Eindringlingen oder das Konvertieren von Video-Streams durchgeführt werden. Im Folgenden werden drei, in ihrer Entwicklung weit fortgeschrittene Anwendungsbeispiele vorgestellt und beschrieben. Alle drei wurden auf der SIGCOMM, einer bedeutenden Konferenz für die Fachgebiete Kommunikation und Computer-Netzwerke, vorgestellt. Sie verdeutlichen noch einmal das große Potential von OpenFlow.

3.2.1 Plug-n-Serve

Plug-n-Serve ist ein dynamisches Lasten-Verteilungssystem auf OpenFlow-Basis. Der Controller steuert den Traffic zu und von den Servern innerhalb eines unstrukturierten Netzwerks. Der Traffic wird dabei nach den Regeln des eigens dafür entwickelten Optimierungs-Algorithmus LOBUS auf die Server verteilt. In der auf der SIGCOMM09 durchgeführten Demonstration und wie in Referenz [4] beschrieben wurde als Beispiel ein Netz von Webservern vorgeführt. Normalerweise würden durch ein statisches Lasten-Verteilungssystem die http-Anfragen gleichmäßig auf alle Server verteilt. Diese Methode kann für Netzwerke, die drauf ausgelegt sind viele Server möglichst gut an das Internet anzubinden, ausreichen, für den Einsatz in unstrukturierten Netzwerken wie Campus-Netzen ist sie allerdings ungeeignet. Plug-n-Serve verteilt im genannten Beispiel die http-Anfrage nicht gleichmäßig, sondern nach den Kriterien der Netz- und Serverauslastung. Die Routing-Entscheidung wird hierbei von dem im Controller implementierten LOBUS-Algorithmus getroffen. Wie in Abbildung 4 zu sehen besteht der Controller hier aus drei wesentlichen Komponenten, dem Net-Manager, der die Auslastung des Netzes überwacht, dem Host-Manager der Informationen über den Zustand der Server sammelt, und dem Flow-Manager der die Routing-Entscheidung für jede Anfrage fällt und dann entsprechend die FlowTables der Switches und Router konfiguriert. Der Vorteil dieses in Referenz [4] beschriebenen Systems ist eine effizientere Nutzung der vorhandenen Ressourcen und eine effektivere Anbindung der Server an das Internet.

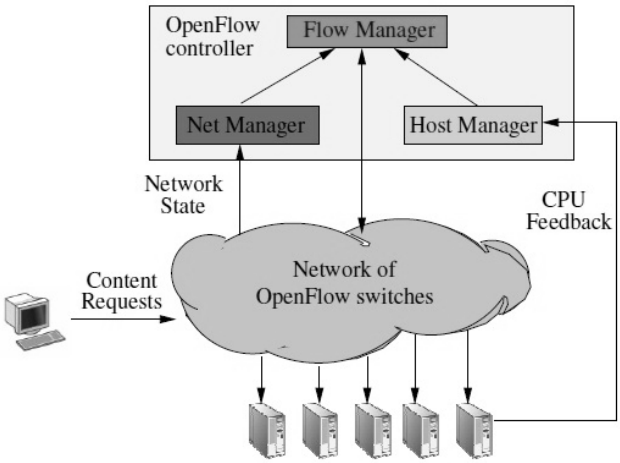

Abbildung 4. Ein Plug-n-Serve Netz [4]

3.2.2 OpenPipes

OpenPipes ist eine OpenFlow-Controller-Software zum Testen von High-Speed Networking Systems. Diese Systeme bestehen aus einer Reihe von Modulen die hintereinander oder parallel geschaltet und von einem Datenstrom durchlaufen werden. Dabei wird der Datenstrom in den einzelnen Modulen analysiert oder bearbeitet. Beispiele dafür sind bestimmte Firewalls oder Video-Bearbeitungs-Systeme. Das Problem bei der Entwicklung dieser Systeme ist, dass Design-Entscheidungen in der Entwurfsphase getroffen werden müssen und sich Tests unter realistischen Bedingungen schwierig gestalten. Referenz [5] bietet dazu mit OpenPipes folgende Lösung am Beispiel eines Video-Bearbeitungs-Systems. Wie in Abbildung 5 zu sehen werden die Video-Daten durch einen Host in ein Open-Flow-Netzwerk geschickt. Die Module befinden sich hierbei auf verschiedenen Computern im Netz und werden je nach Einstellung des Controllers durch verschiedene Module geleitet. Dabei lässt sich die Reihenfolge ändern um verschiedene Konfigurationen des Systems zu testen. Die Abbildung 5 zeigt die GUI der Controller-Software. Dabei symbolisieren die gestrichelten Rechtecke die Rechner im Netz, die Buchstaben die einzelnen Module und die Pfeile die Verbindungen, die OpenFlow zwischen ihnen herstellt. OpenPipes ist ein weiteres Beispiel dafür, was für vielfältige Anwendungsmöglichkeiten OpenFlow ermöglicht.

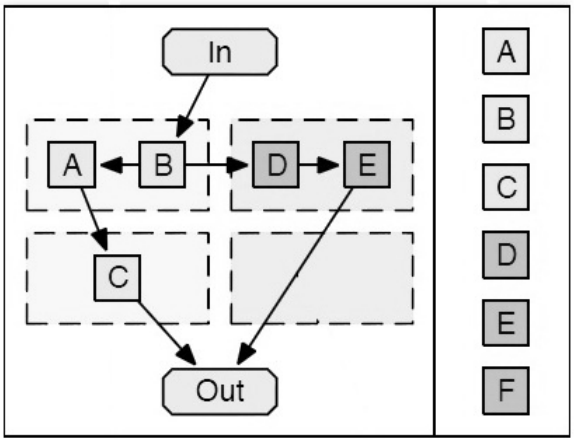

Abbildung 5. GUI der OpenPipes-Controller-Software [5]

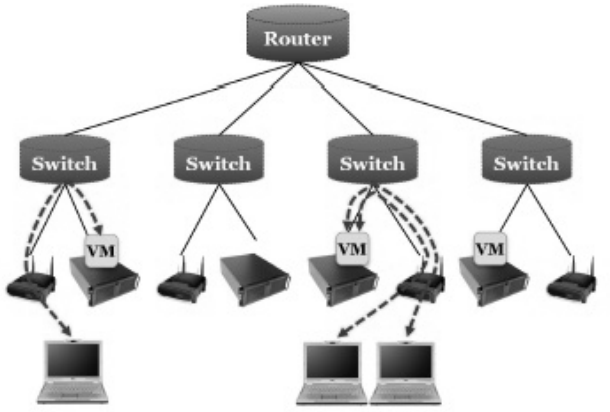

Abbildung 6. Ein OpenFlow-Netzwerk mit Virtuellen Maschinen [6]

3.2.3 Virtual Machine Mobility in OpenFlow

Unter Virtual Machine Mobility versteht man die Möglichkeit virtuelle Maschinen und deren Betriebssysteme im laufenden Betrieb von einem Rechner auf den anderen zu verlagern. So kann beispielsweise ein virtueller Spiele-Server auf einen anderen Computer verschoben werden ohne dass der Server neu gestartet werden muss. Das Problem hierbei ist allerdings, dass alle Clients, die eine Verbindung mit dem Server unterhalten, diese Verbindung verlieren, da der Server nach der Verlagerung nur mehr unter seiner neuen IP-Adresse zu erreichen ist. Die Referenz [6] beschreibt eine Lösung für dieses Problem. Auf der SIGCOMM08 wurde anhand dieses Beispiels demonstriert, wie OpenFlow es ermöglicht den Spiele-Server zu verschieben und gleichzeitig die Verbindung der Clients nicht abreißen zu lassen. Hierzu wurden mehrere OpenFlow-Netzwerke innerhalb der USA und Japan durch IP-Tunnel miteinander verbunden um ein größeres OpenFlow-Netz zu schaffen. An einem der Standorte lief der Spiele-Server, während die Clients von anderen Standorten aus zu dem Server eine Verbindung aufbauten. Auch hier wurde der virtuelle Spiele-Server auf einen anderen Rechner in einem anderen Netz verlagert. Die Verbindung zu den Clients blieb dabei allerdings erhalten. Dies ermöglichte der Controller indem er die Datenströme zwischen Server und den Clients auf die neue Position umlenkte. Die von den Clients versandten Pakete mit der nicht mehr aktuellen Zieladresse des Servers wurden vom Controller auf die neue Adresse umgeleitet. Diese Technik funktioniert auch wenn nicht der Server sondern der Client seinen Zugang zum Netzwerk wechselt, also sich zum Beispiel mit einem anderen Access Point verbindet. Dieser problemlose Positionswechsel der Netz-Teilnehmer eröffnet weitere interessante Möglichkeiten. So könnte ein Server durchgehend nach der Position im Netz suchen bei der er die optimale Verbindung zu seinen Clients schaffen kann um dann auf diese Position zu wechseln. Dieses Beispiel veranschaulicht noch einmal welches Potential in OpenFlow steckt und welchen Einfluss es auf die zukünftige Entwicklung von Netzwerken-Technologien haben kann.

4. Das OpenFlow Projekt

OpenFlow wurde an der Stanford Universität entwickelt. Mit der Entwicklung wurde 2007 begonnen und seitdem wird es durchgehend weiterentwickelt. Die Spezifikation [2] wurde 2009 in der Version 1.0.0 veröffentlicht. Das OpenFlow-Konsortium besteht größtenteils aus Forschern und Netzwerk-Administratoren. Jedoch kann jeder der die fachliche Kompetenz besitzt und sich einbringen will beitreten. Ihr Ziel ist es die Spezifikation zu verbessern und zu erweitern. OpenFlow ist für kommerzielle und nicht-kommerzielle Produkte frei nutzbar. Die Referenz [7] listet die wichtigsten Unterstützer der OpenFlow-Entwickler auf. Dazu zählen namhafte Firmen wie Cisco, HP, NEC, T-Mobile und Ericsson, die das OpenFlow-Projekt mit finanziellen Mitteln und die Bereitstellung von Hardware fördern. Viele Hersteller implementieren bereits seit mehreren Jahren das OpenFlow-Protokoll in ihre Geräte. Eine ganze Reihe an Forschungseinrichtungen und Universitäten konnten OpenFlow dadurch schon in ihre Netz-Infrastrukturen integrieren

5. ZUSAMMENFASSUNG

Obwohl OpenFlow erst vor wenigen Jahren entwickelt wurde kommt es schon verstärkt zum Einsatz. Die hier beschrieben Anwendungsbeispiele zeigen ein Teil der vielen Ideen die in OpenFlow-Netzen umgesetzt wurden und noch werden. Daher erscheint OpenFlow als eine vielversprechende Technologie. Das Magazin MIT Technology Review [7] zählt das OpenFlow-Protokoll zu einer wichtigen technologischen Neuentwicklung. Kritisch ist anzumerken, dass dieses Protokoll nicht ohne weiteres in Netzen einzusetzen ist, die die Größe von Universitäts-Netzwerken übersteigen, denn hier drängt sich die Frage auf wer die Macht besitzen darf den Controller zu kontrollieren. Der wohl größte Nachteil von OpenFlow ist allerdings die Tatsache, dass die in einem normalen Netzwerk eingesetzten Geräte durch OpenFlow-fähige Hardware ersetzt werden müssen, um überhaupt ein OpenFlow-Netzwerk aufbauen zu können. Nichtsdestoweniger wird die weitere Verbreitung von OpenFlow die Forschung auf dem Gebiet der Netzwerk-Technik unterstützen und positiv beeinflussen.

6. REFERENCES

[1] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. und Turner, J.. OpenFlow: Enabling Innovation in Campus Networks, März 2008

[2] OpenFlow Switch Specification, Version1.0.0, Dez 2009, http://www.openflowswitch.org/wp/documents/

[3] Sherwood, R., Chan, M., Gibb, G., Handigol, N., Huang, T.-Y., Kazemian, P., Kobayashi, M., Underhill, D., Yap, K.-K., Appenzeller, G. und McKeown, N.. Carving Research Slices Out of Your Produciton Networks with OpenFlow, SIGCOMM 2009

[4] Handigol, N., Seetharaman, S., Flajslik, M., McKeown, N. und Ramesh, J., Plung-n-Serve: Load-Balancing Web Traffic using OpenFlow, SIGCOMM2009

[5] Gibb, G., Underhill, D., Covington, A., Yabe, T. und McKeown, N., OpenPipes: Prototyping high-speed networking systems, SIGCOMM2009

[6] Erickson, D., Gibb, G., Heller, B., Underhill, D., Naous, J., Appenzeller, G., Parulkar, G., McKewon, N., Rosenblum, M., Lam, M., Kumar, S., Alaria, V., Monclus, R., Bonomi, F., Tourrilhes, J., Yalagandula, P., Banerjee, S., Clark, C. und McGeer, R., A Demonstration of Virtual Machine Mobility in an OpenFlow Network, SIGCOMM 2008

[7] K. Greene. Special reports 10 emerging technologies 2009.MIT Technology Review, 2009. http://www.technologyreview.com/biotech/22120/