Overlay Convergence Architecture for Legacy Applications (OCALA)

Dieter Panin
Betreuer: Heiko Niedermayer
Seminar Future Internet WS2010/2011
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: panind@in.tum.de

KURZFASSUNG

Die Umstellung von der IP-basierten Architektur des Internets auf die eines Overlaynetzwerks war für die User stets mit Einstellungen am Betriebssystem, sowie an den auszuführenden Anwendungen verbunden. Um aber dem Nutzer die Vorteile der Overlays näher zu bringen, muss es möglich sein, ein Overlay bequem und ohne nötige Voreinstellungen laufen zu lassen. Dieses Ziel setzten sich die Entwickler von OCALA[1][2], einer Netzarchitektur, die eine automatische Abstimmung zwischen dem Overlay und IP, sowie mehreren Overlays untereinander ermöglicht. OCALA fügt dazu eine neue Schicht in das ISO/OSI-Modell ein, die Overlaykonvergenzschicht OC. Diese befindet sich unter der Transportschicht und besteht aus zwei Subschichten, der overlayunabhängigen OC-I, die eine Schnittstelle nach oben für die Anwendung bereitstellt, und der overlayabhängigen OC-D, die mit der darunter liegenden Schicht Daten austauscht.

SCHLÜSSELWORTE

Network Architecture, Netzarchitektur: Ein Konzept, das den Aufbau von und die Interaktion in Netzwerken beschreibt.

Overlaynetzwerk (kurz: Overlay): Eine bestimmte Form von Netzarchitektur, die auf einem vorhandenen Netzwerk, meist dem Internet, basiert. Overlays sind logische Netzwerke und verfügen meist über einen eigenen Adressraum mit eigener Adressierung. Ein bekanntes Beispiel für Overlays sind Peer-to-Peer-Netzwerke. In dieser Arbeit sollen aber nur die Overlays RON[3][4] und i3[5] näher betrachtet werden. (Das Internet selbst ist übrigens ein Overlay des Telefonnetzwerks.)

Legacy Application: Im Rahmen dieser Arbeit sind damit alle bekannten Netzwerkanwendungen gemeint, die über die IP-Schicht kommunizieren. Allgemein spricht man von einer „vererbten Anwendung“ bei Software, die bereits vor einer entscheidenden Neuerung im Gesamtkonzept ausgeführt wurde.

1. EINLEITUNG

Ein Hauptgebiet der aktuellen Netzwerkforschung ist die Aufhebung der funktionalen Grenzen, die das Internet hat. Ein in dieser Richtung oft eingeschlagener Ansatz ist die Entwicklung von alternativen Netzarchitekturen und als Spezialform von diesen, die der Overlaynetzwerke. Diese bieten Möglichkeiten wie ein effizienteres Routing der Datenpakete, Anonymität, die Adressierung von Hosts hinter einem NAT, die immer wichtiger werdende Mobility und viele weitere. So entstand über die letzten zwei Jahrzehnte eine Vielzahl an Vorschlägen und Realisierungen dieser Architekturen, die jedoch zum Großteil nicht in der Praxis getestet werden konnten. Ein Grund dafür ist die Schwierigkeit der Endnutzer, sich mit den nötigen Einstellungen auseinanderzusetzen, die zur letztendlichen Ausführung und Evaluierung der Overlays nötig wären. Somit ist eine Architektur erforderlich, die dem Nutzer für all seine Anwendungen von Browser bis Remote Desktop diese Arbeit abnimmt und diese mit dem Overlay kommunizieren lässt, als ob ihnen die sonst erwartete IP-Adressierung und DNS Name Resolution zugrunde liegen würden. Denn Overlays können sich anderer Schnittstellen bedienen, die diese Funktionen erfüllen. Hierzu gab es Lösungen, die sich auf ein bestimmtes Overlay bezogen, jedoch noch keine, die für alle Overlays funktioniert.

Diese Aufgabe soll OCALA, die Overlay Convergence Architecture for Legacy Applications lösen und damit einen einfachen Zugriff auf die nützlichen Funktionen, die Overlays bieten, ermöglichen. Im Rahmen der Implementierung sollen dabei vier Anforderungen erfüllt werden: (1) Anwendungen, die auf einem Host laufen, sollen gleichzeitig auf verschiedene Overlays zugreifen können, (2) die Verbindung mehrerer unterschiedlicher Overlays, sodass Nutzer in verschiedenen Overlays miteinander kommunizieren können, (3) die Möglichkeit, zwei Hosts sogar dann durch ein Overlay kommunizieren zu lassen, wenn einer von ihnen nur IP versteht und (4) Erweiterbarkeit, also die Möglichkeit, ein neues Overlay mit minimalem Aufwand in OCALA zu integrieren.

Diese Anforderungen unterscheiden OCALA von anderen Systemen wie Oasis[6], einem Projekt auf dem Gebiet der Overlays, das Anwendungen ihren Traffic über Overlays routen lässt. Einige andere Konzepte konzentrieren sich hingegen auf die Implementierung von Verknüpfungen von Overlays, also die mögliche Interaktion von Hosts in zwei verschiedenen Overlays untereinander. OCALAs Aufgabe soll es hingegen vor allem sein, die Benutzer einfach auf Overlays zugreifen zu lassen und ihnen so die damit verbundenen Vorteile näher zu bringen.

doi: 10.2313/NET-2011-05-2_04

Das Konzept von OCALA sieht dazu vor, die Architektur als eine Proxy zu implementieren, die unter der Transportschicht eine Overlaykonvergenzschicht einfügt. (Ihre genaue Funktionalität wird in Abschnitt 2.2 behandelt.) Die Implementierung als Proxy bietet dabei den Vorteil, dass für OCALA selbst wiederum auch keine Änderungen, weder an Anwendungen noch am Betriebssystem, nötig sind. Ferner sollen in dieser Arbeit die Möglichkeiten und Grenzen von OCALA aufgezeigt und erläutert, sowie eine genauere Darstellung der eigentlichen Funktionsweise gegeben werden.

2. FUNKTIONALITÄT ALLGEMEIN

Da OCALA unter der Transportschicht arbeitet, bietet es keine Unterstützung für Overlays der Transport- und Anwendungsschicht. Das Hauptaugenmerk liegt auf Overlays, die einen End-to-End-Pakettransfer nach dem Muster von IP unterstützen. Diese Overlays haben die Möglichkeiten, die Leistung des Internets zu erhöhen, neue Funktionen wie *Mobility*[7] zu implementieren oder, wie i3, ihre architekturspezifischen Vorteile zu bieten.

Wie schon angedeutet, ist die Adressierung in einem Overlay vollkommen den Entwicklern überlassen. Die einzige Beschränkung ist, dass jeder Endhost über einen eindeutigen Identifier (*ID*) in seinem Overlay erreichbar ist. Dazu kann, wie im Fall von RON, die IP-Adresse des Hosts verwendet werden. Andere Overlays bedienen sich jedoch eigener IDs, wie Hashes von Public Keys. Zusätzlich können zur Vereinfachung der Adressierung von Hand für Menschen besser lesbare Namen den IDs zugeordnet werden.

2.1 Die vier Ziele

OCALAs Aufbau und Funktionsweise lassen sich am besten mit den vier Zielen beschreiben, die sich die Entwickler gesetzt haben. Diese vier Ziele sind:

- **Transparenz**: Die Anwendungen sollen fehlerfrei funktionieren, auch wenn sie über einem Overlay an Stelle von IP laufen.
- **Überbrückung verschiedener Overlays**: Hosts in verschiedenen Overlays sollen Daten austauschen können und Hosts, die kein Overlay verwenden, sollen aus einem Overlay heraus erreichbar sein.
- **Funktionalität der Overlays offenlegen**: Die Nutzer sollen frei wählen können, welche Overlays und mit ihnen verbundenen Funktionalitäten sie für ihre Anwendungen einsetzen wollen.
- **Gemeinsame Funktionen bereitstellen**: Wichtige Funktionen, wie die auf dem Gebiet der Security, sollen nicht von Overlays übernommen, sondern in OCALA implementiert werden.

2.2 Overlaykonvergenzschicht

Wie bereits erwähnt, implementiert OCALA eine neue Schicht in das ISO/OSI-Modell. Diese befindet sich aus der Sicht der Anwendung an Stelle der Netzwerkschicht. Dies ist nötig, da das Overlay die sonst vorhandene Netzwerkschicht ersetzt. Diese Overlaykonvergenzschicht (siehe Abbildung 1), kurz OC, besteht aus zwei Subschichten, der overlayunabhängigen OC-I und der overlayabhängigen OC-D.

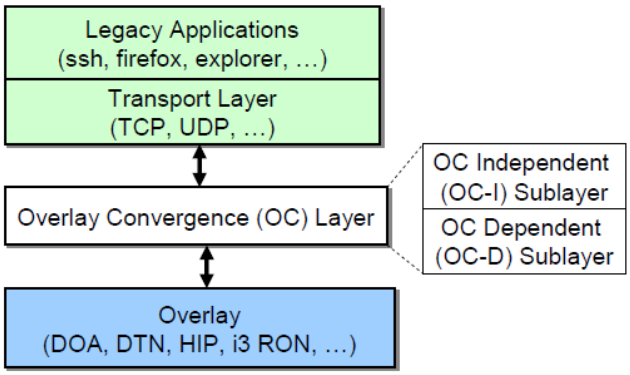

Abb 1. Lage und Aufbau der OC im Protokollschichtenstack[1]

Die OC-I simuliert für die Anwendung die IP-Schicht, nimmt von ihr also Daten entgegen. Außerdem ordnet sie die Daten den einzelnen Anwendungen und den von ihnen verwendeten Overlays zu und implementiert ein paar wichtige Funktionen, wie Authentifizierung und Verschlüsselung.

Die OC-D ihrerseits besteht aus einzelnen overlayspezifischen Modulen, die für das jeweilige Overlay eine Verbindung aufbauen und Pakete verschicken und empfangen können. In diesem Zusammenhang kann IP als ein Default Overlay gesehen werden, an das die Daten gehen, wenn kein eigentliches Overlay verwendet wird.

Abbildung 2 zeigt die OC in Aktion, ein Host A führt 3 Programme aus, mit Firefox greift er über das Internet auf *cnn.com* zu, über i3 chattet er anonym in IRC und lässt seine ssh-Daten mit RON effizienter routen. Dabei sieht man, dass Hosts B und C auch OCALA ausführen, da auch bei ihnen eine OC vorhanden ist. Dies ist nicht zwingend nötig, da OCALA Hosts auch über verschiedene Overlays hinweg kommunizieren lässt.

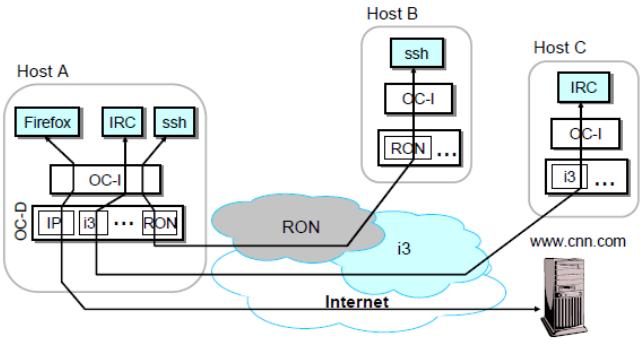

Abb. 2: Drei Verbindungen über drei verschiedene OC-D-Module[1]

Wie ein Host A über i3 einen Host C erreichen kann, der RON nutzt, zeigt Abbildung 3. Dazu wird ein Host B dazwischengeschaltet, dessen OC den Traffic aus i3 nach RON leitet. Man sieht, dass dazu aus dem i3-Modul der OC-D die Daten an die OC-I gehen. Der ganze Übergang von einem Overlay in ein

doi: 10.2313/NET-2011-05-2_04

anderes ist also nur ein zusätzlicher Hop auf der Route zwischen A und C.

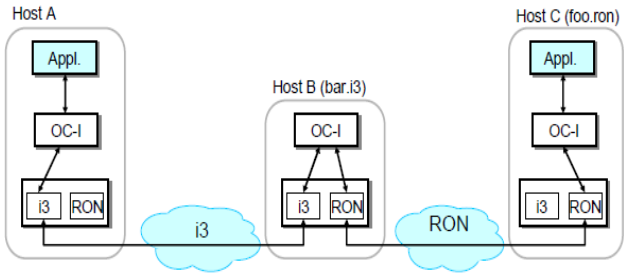

Abb. 3: Übergang aus einem Overlay in ein anderes[1]

Der Kommunikationskanal zwischen zwei Endhosts auf der OC-I wird dabei Pfad und der auf der OC-D Tunnel genannt. Im Beispiel aus Abbildung 3 ist der Pfad A,B,C und besteht aus den beiden Tunneln A,B und B,C.

Damit lässt sich eine Parallele zur Funktionalität der zweiten und dritten Schicht des ISO/OSI-Modells ziehen, denn so wie die niedriger liegende Sicherungsschicht (*data link layer*) die Verbindung zwischen zwei Knoten in einer Link Layer Domain implementiert, implementiert die OC-D-Subschicht die Kommunikation zwischen zwei Knoten in einem Overlay. Und wie die höhere Vermittlungsschicht (*network layer*) eine Kommunikation zwischen verschiedenen Link Layer Domains ermöglicht, ermöglicht die OC-I die Kommunikation zwischen verschiedenen Overlays.

3. GENAUE FUNKTIONSWEISE

Die Funktionsweise von OCALA lässt sich am besten mit der Realisierung der in Abschnitt 2.1 erwähnten Ziele beschreiben.

3.1 Transparenz

Um einen fehlerfreien Einsatz der Anwendungen zu ermöglichen, hat man an ihrem eigentlichen Funktionsschema angeknüpft. Dieses ist meist so, dass eine DNS-Anfrage gestellt wird und dann IP-Pakete mit der zurück erhaltenen IP-Adresse ausgetauscht werden. Allgemein haben Anwendungen zwei Möglichkeiten der Adressierung, entweder die beschriebene, wobei der DNS-Anfrage ein Name übergeben wird oder der direkte Weg über die Angabe einer IP-Adresse.

Ausgehend von diesem Konzept bietet auch OCALA zwei Möglichkeiten der Adressierung von Overlays an.

Die erste Möglichkeit ist die, den Benutzer Regeln festlegen zu lassen, wie Pakete anhand ihrer IP-Header zu versenden sind. So kann man Pakete für *64.236.24.4 Port 80* über RON und Pakete an *207.188.7.x* über i3 leiten lassen. Der Vorteil dieser Möglichkeit liegt dabei darin, dass es für jede Anwendung funktioniert, da jede (hier relevante) Anwendung IP-Pakete austauscht.

Die zweite Möglichkeit ist die, das zu verwendende Overlay im DNS-Namen anzugeben. Dazu gibt der Benutzer einen Namen der Form *bsp.ov* an, wobei *bsp* der (eindeutige) Name eines Hosts in dem Overlay *ov* ist. Erhält die OC eine Anfrage dieser Form, baut sie eine Verbindung zu *ov* auf und leitet den Traffic der sendenden Anwendung an *bsp.ov* weiter. Diese Möglichkeit bietet gleich mehrere Vorteile. Erstens können so Hosts adressiert werden, die keine eigene IP besitzen, was zum Beispiel bei Hosts hinter NATs der Fall ist. Zweitens sind die Namen für Menschen lesbarer und damit praktischer im Gebrauch als IPs. Drittens braucht der User die IP nicht zu kennen, denn oftmals kann er sie (aus verschiedenen Gründen) auch nicht kennen.

3.1.1 End-to-End Pfadaufbau

Ein Verbindungsaufbau kann durch zwei Umstände ausgelöst werden, erstens: eine DNS-Anfrage für einen zuvor unbekannten Overlayhost wird erhalten oder zweitens: den Empfang eines ersten Datenpakets aus einem Overlay. Am Ende des Verbindungsaufbaus soll ein End-to-End-Pfad zwischen den OC-I-Schichten der beiden Kommunikationspartner bestehen. Der Einfachheit halber wird im folgenden Beispiel nur ein Tunnel dafür verwendet.

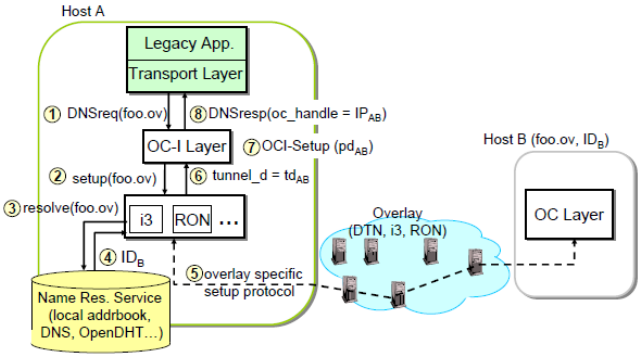

Abb. 4: Der Pfadaufbau in seinen Einzelschritten[1]

Abbildung 4 zeigt, wie eine Anwendung auf Host A mit einer Anwendung auf B, mit dem Namen *foo.ov* kommunizieren will. Zunächst wird eine DNS-Anfrage für *foo.ov* gestellt (1), welche die OC-I abfängt. Dabei erstellt die OC-I nebenbei einen *path descriptor* (*pd*), der den Pfad später eindeutig identifizieren soll. (Ein 128-Bit-Feld zur Kollisionsvermeidung.) Dann schickt die OC-I den Befehl, an das *ov*-Modul der OC-D, einen Tunnel zu foo.ov aufzubauen (2). Die OC-D ihrerseits benutzt den individuellen Name Resolution Service des Moduls um *foo.ov* zu finden (3). (Zur Erinnerung: diese Resolution Services können DNS oder ein overlayinterner Dienst sein, wie das Namenshashing in i3.) Der Name Res. Service liefert die zu *foo* gehörende ID an die OC-D zurück (4). Die OC-D hat nun alle Informationen, um eine Verbindung zu B aufzubauen. Sie tut dies mit einem für das Overlay definierten Protokoll (5). Ist der für den Datenaustausch benötigte Status erreicht, wird ein Pointer auf diesen Status, der *tunnel descriptor* (*td*) an die OC-I übergeben (6). Nach dem Empfang des Tunnel Descriptors führt die OC-I von A zusammen mit der OC-I von B einen OCI-Setup durch (7). Ist dies abgeschlossen, so übergibt die OC-I der Anwendung einen *oc_handle* in Form einer DNS-Response. Das *oc_handle* ist eine local-scope-IP, die der Verbindung mit foo.ov zugewiesen wird. Diese Adresse wird die Anwendung nun für den Paketaustausch mit foo.ov verwenden. Der vergebene IP-Adressenbereich beschränkt sich dabei auf *1.x.x.x*. Tunnel descriptors werden zwischen der OC-I und OC-D eines Hosts ausgetauscht, path descriptors zwischen den OC-Is verschiedener Hosts. Die Trennung von Tunnel und Path Deskriptoren hat den Vorteil, dass verschiedene paths denselben tunnel descriptor nutzen können. Z.B. Pfade (A,B,C) und (A,B,D) nutzen Tunnel (A,B).

doi: 10.2313/NET-2011-05-2_04

3.2 Überbrückung von Overlays

Es gibt einige Erreichbarkeitsprobleme, die Overlays betreffen, denn diese verfügen über sehr unterschiedliche Funktionalitäten. So können Hosts hinter einem NAT zwar über i3, jedoch nicht über RON erreicht werden, da i3 jedem Host eine ID zuordet während Hosts in RON über ihre IP-Adresse identifiziert werden. Außerdem sollen auch Hosts aus einem Overlay heraus erreicht werden können, die nur das Internet nutzen.

OCALA löst diese Probleme mit dem Konzept der *remote resolution*. Diese funktioniert so, dass wenn ein Host im Overlay ov1 eine Anfrage für *bsp.ov2* sendet, diese Anfrage an ein Gateway geleitet wird, das *ov2* ausführt. Dort geht sie aus dem *ov1*-Modul der OC-D an die OC-I und von dort wieder an die OC-D, jedoch an das zuständige *ov2*-Modul, wo sie aufgelöst wird.

Dass ein Host, der OCALA ausführt, zwar ein Overlay nutzen kann, aber auch die Möglichkeit hat, seine Anwendungen weiterhin direkt über IP kommunizieren zu lassen, ist bereits bekannt. Dazu wird der Traffic lediglich über das IP-Modul der OC-D geleitet. Was aber, wenn er nur aus einem Overlay heraus auf einen Internethost zugreifen möchte? Oder, was vielleicht noch wichtiger ist, was haben die Leute von OCALA, die nur über das Internet kommunizieren wollen? Diese Fragen löst OCALA mit dem Konzept der *legacy gateways*.

3.2.1 Legacy gateways

Legays gateways haben Ähnlichkeit mit Overlay gateways (vgl. Abbildung 3), wobei einer der beiden Tunnel über IP verläuft. Die Funktionalität des Overlays gilt für diesen Tunnel natürlich nicht. Es lassen sich zwei Arten von Legacy Gateways unterscheiden.

Das legacy server gateway ermöglicht die Verbindung eines Overlay clients mit einem *legacy server*. Er bekommt die Daten vom OC-D-Modul des Overlays, das der Host verwendet, übergibt sie der OC-I-Subschicht und leitet sie von dort an das sogenannte *LegacyServerIP*-Modul. Das legacy gateway verhält sich in diesem Fall wie ein NAT des *overlay clients*, es weist der Verbindung einen seiner lokalen Ports zu und ändert entsprechend die Adressen im IP-Header. (siehe Abbildung 5).

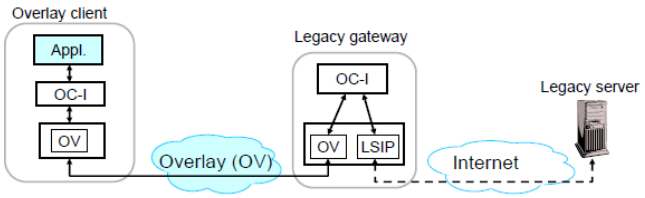

Abb. 5: Legacy server gateway in Aktion[1]

Das legacy client gateway lässt Hosts, die sich weder in einem Overlay befinden, noch über eine OC-Schicht verfügen, eine Verbindung zu einem Overlay herstellen. Der Traffic vom *legacy client* wird mit einem *LegacyClientIP*-Modul empfangen, zur OC-I geleitet und von dort aus zum gewünschten Overlaymodul der OC-D transferiert. Sobald die Verbindung aufgebaut wurde, schickt das legacy gateway eine DNS-Response mit einer IP-Adresse an den Client. Immer, wenn der Client nun IP-Pakete mit dieser IP schickt, werden sie an den *overlay server* übermittelt (siehe Abbildung 6). Ein bestehendes Problem dieser Methode ist aber, dass die im DNS-Response enthaltene IP-Adresse routbar sein muss. Dies beschränkt die Anzahl der möglichen Verbindungen mit dem legacy gateway.

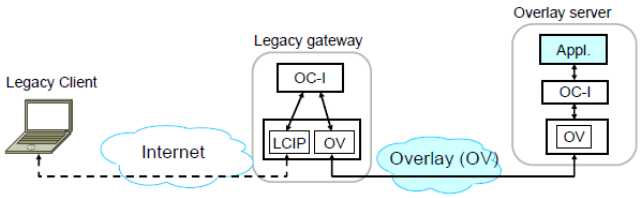

Abb. 6: Legacy client gateway in Aktion[1]

3.3 Overlayfunktionalität offenlegen

Jedes Overlay bietet seine eigenen, individuellen Möglichkeiten zur Verbesserung der Datenübertragung. In RON kann man z.B. selbst die Routingmetriken für Pakete angeben und mit i3 kann man sogenannte Middleboxes in einen Pfad einfügen. Der Nutzer von OCALA soll dabei nach eigenem Ermessen für jeden Tunnel entscheiden können, welche Optionen für ihn sinnvoll sind und auf welche er verzichten möchte.

Der erste Ansatz, dies möglich zu machen, war, in die DNS-Namen eine Syntax der Form *bsp.delay50ms.overqos* einzufügen. Dieser Befehl bewirkt eine Verbindung mit weniger als 50 Millisekunden Delay an den Host mit dem Namen *bsp* im *OverQoS*-Overlay[8]. Diese Idee hat sich aber in der Praxis als unhandlich herausgestellt und wurde, trotz ihrer bestehenden Implementierung in OCALA, nicht verwendet. Stattdessen wurde eine GUI erstellt, in der der Benutzer vor dem Verbindungsaufbau die overlayspezifischen Optionen festlegen kann. Diese GUI schreibt entsprechende Einträge in dafür bestimmte Konfigurationsdateien im XML-Format. Der User kann aber auch direkt auf diese zugreifen und Änderungen an ihnen vornehmen.

3.4 Gemeinsame Funktionen bereitstellen

Um die Arbeit der Overlayentwickler zu erleichtern, implementiert OCALA auf der OC-I-Subschicht generische Funktionen, die für alle Overlays nützlich sind. Dazu zählen derzeit Securityfunktionen und Datenkomprimierung.

Die Securityfunktionen bieten Verschlüsselung von Daten und Authentifizierung von Verbindungen. Beide beruhen auf der Annahme des Vorhandenseins einer zentralen Stelle für Zertifizierung und Namenszuweisung im Overlay. Das Protokoll, das OCALA dafür verwendet, ist dem von SSL ähnlich.

4. OCALAS PROBLEME UND MÖGLICHKEITEN

Die Ausführung von OCALA bringt neben vielen Vorteilen auch ein paar Probleme mit sich. Im Folgenden sollen diese beschrieben und anschließend mit den einzelnen Möglichkeiten, die OCALA bietet, aufgewogen werden.

4.1 Probleme

Zunächst werden Overlays der Transport- und Anwendungsschicht auch in Zukunft nicht unter OCALA laufen, da diese sich über der OC einfügen. Diese Overlays können aber dem Nutzer einige nützliche Funktionen bieten.

Intuitiv drängt sich auch die Frage nach dem mit OCALA verbundenen Rechenaufwand auf. Wie die Tests der Entwickler

doi: 10.2313/NET-2011-05-2_04

zeigten, sind die mit dem zusätzlichen Overhead verbundenen Kosten zwar bei lokalen Netzwerken gering, können aber bei Fernnetzen schnell ansteigen. Dies hängt aber auch mit der Ausprägung der Overlaynetzwerke zusammen.

Ferner unterstützt OCALA derzeit nur Unicast-Anwendungen. Anwendungen, die sich Multicast bedienen, laufen unter OCALA nicht. Dies ist jedoch eine Aufgabe, an der die Entwickler von OCALA noch arbeiten werden[1].

Wie schon bekannt, liefert die OC-I lokale IP-Adressen. Die damit verbundenen Probleme sind erstens, dass diese Adressen auf anderen Hosts nicht erreichbar wären. Zweitens würden Anwendungen, die ftp benutzen, welches IP-Adressen in Datenpaketen verschlüsselt, nicht mit OCALA vereinbar sein, da die OC-I die IP-Header vor der Übermittlung überschreibt. Obwohl dies in einem Overlay durch gemeinsame Verhandlung über die der Anwendung übergebenen IP-Adresse verhindert werden kann, werden beim Übergang eines legacy gateways stets die Adressen überschrieben.

4.2 Möglichkeiten

NAT Traversal: Da sich über i3 auch Maschinen hinter NATs erreichen lassen, ist es mit dem i3-Modul der OC-D möglich, auch einen legacy server hinter einem NAT aufzustellen. Außerdem kann man sich so bequem von überall aus mit dem Heimcomputer verbinden, indem man seinen i3-Namen adressiert.

Middleboxes: In i3 kann man angeben, ob man eine direkte Verbindung wünscht oder über eine Middlebox, die als zusätzlicher Hop auf dem Weg zum Empfänger eingefügt wird, kommunizieren will. Auf diesen Middleboxes lassen sich Anwendungen ausführen, die Funktionen wie *intrusion detection* für die Verbindung ermöglichen.

Abgesicherte Mobility: Mit HIP[9] lässt sich eine abgesicherte Verbindung zwischen zwei mobilen Hosts herstellen. Dieses Feature nutzt OCALA, um eine *ssh*-Verbindung selbst dann aufrecht zu erhalten, wenn einer der Hosts seine IP ändert.

Sicherheit und Flexibilität in VPNs: Die Entwickler von OCALA bieten einen Vorschlag zur Verbesserung der Sicherheit und Flexibilität in VPNs. Dazu wird intern ein legacy server gateway zwischengeschaltet, durch das alle Daten von externen Hosts (aus Overlays) an interne geleitet werden. Die Sicherheitsfunktionen der OC-I spielen dabei eine wichtige Rolle. Damit kann der Benutzer gleichzeitig auf mehrere Intranets zugreifen, selbst wenn sie alle denselben Adressraum verwenden. Ein weiterer Vorteil ist, dass der externe Host keine interne IP-Adresse zugewiesen bekommt. Das erschwert Scanangriffe auf das interne Netz.

Overlay Composition: Ohne OCALA kann der User lediglich ein Overlay verwenden, so ist es für ihn nicht möglich, die Funktionalitäten verschiedener Overlays miteinander zu kombinieren. Durch OCALA ist dies mit der Zwischenschaltung von Overlaygateways möglich. Der User kann also mit Hilfe von i3 ohne Verbindungsverlust zwischen verschiedenen drahtlosen Netzwerken wechseln und die Routingentscheidungen dabei von RON optimieren lassen.

5. ZUSAMMENFASSUNG

Die Overlays stellen mit Sicherheit eine Verbesserung im Hinblick auf die Funktionen des normalen Internets dar und können deshalb in Zukunft mehr an Bedeutung gewinnen. Dank OCALA können sie, wenn es so weit ist, auch den Endnutzern zugänglich gemacht werden. Ein aktueller Trend in der Entwicklung des Internets lässt sich aber leider nur schwer erkennen. Seit Jahren wird von der Umstellung von IPv4 auf IPv6 gesprochen, die auch weiterhin auszubleiben scheint. Mit Hilfe von Overlays könnte aber genauso gut der Adressraum von IPv4 erweitert werden. Dies ist für mich ein Zeichen dafür, dass das Potential, welches Overlays mit sich bringen, weitgehend unterschätzt wird.

6. LITERATUR

[1] Dilip Joseph, Jayanthkumar Kannan, Ayumu Kubota, Karthik Lakshminarayanan, Ion Stoica, Klaus Wehrle: "*OCALA: An Architecture for Supporting Legacy Applications over Overlays*", San Jose, CA, USA, Mai, 2006.

[2] OCALA, online: http://ocala.cs.berkeley.edu/

[3] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. *Resilient Overlay Networks*. In *Proc. of SOSP*, 2001.

[4] Resilient Overlay Networks, online: http://nms.lcs.mit.edu/ron/

[5] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. *Internet Indirection Infrastructure*. In *SIGCOMM*, 2002.

[6] H. V. Madhyastha, A. Venkataramani, A. Krishnamurthy, and T. Anderson. Oasis: *An Overlay-aware Network Stack*. *SIGOPS Operating Systems Review*, 40(1), 2006.

[7] P. Yalagandula, A. Garg, M. Dahlin, L. Alvisi, and H. Vin. *Transparent Mobility with Minimal Infrastructure*. Technical Report TR-01-30, UT Austin, June 2001.

[8] dL. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz. *OverQoS: An Overlay-based Architecture for Enhancing Internet QoS*. In *Proc. of NSDI*, 2004.

[9] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. *Host Identity Protocol*, 2003. http://www.hip4inter.net/documentation/drafts/draft-moskowitz-hip-08.html

doi: 10.2313/NET-2011-05-2_04