Integration des Overlay-Multicast-Dienstes Ariba/MCPO in die hybride Multicast-Plattform H∀Mcast

Nora Berg
Nora.Berg@haw-hamburg.de

31. Januar 2013

Zusammenfassung

Für Anwendungsprogrammierer ist es im derzeitigen Internet unattraktiv, Multicast-Anwendungen zu entwickeln, da diese in vielen Fällen - bedingt durch Protokollheterogenität, Mobiltität und Middleboxes - nur mit Einschränkungen funktionieren. H∀Mcast und Ariba/MCPO sind zwei Future-Internet-Ansätze, welche den Zugriff auf Multicast-Technologien vereinfachen und eine möglichst große Anzahl von Empfängern trotz der genannten Schwierigkeiten erreichen.
In dieser Arbeit wird ein Adapter vorgestellt, der Ariba/MCPO in die H∀Mcast-Middleware integriert, um deren Vorteile zu kombinieren. Es werden die Konzepte von H∀Mcast und Ariba/MCPO beschrieben und die Performance der einzelnen Komponenten gemessen.

1 Einleitung

Es gibt im derzeitigen Internet viele Anwendungen, wie Internetradio, IPTV oder Online-Computerspiele (MMOGs), die Daten von wenigen Sendern vielen Empfängern zustellen. Im Laufe der Zeit haben sich viele Gruppenkommunikationsprotokolle auf (fast) allen logischen Ebenen des Netzwerkstacks etabliert. Beispielsweise IP-Multicast, verschiedene Overlaynetzwerke, oder komplette Lösungen, die z.B. von IPTV - Providern selbstentwickelt und vertrieben werden. Das Entwickeln von Anwendungen in dieser heterogenen Multicast-Landschaft ist aufwendig, da verschiedene Domänen meist unterschiedliche oder keine Multicastprotokolle unterstützen. Diese Domänen bilden Multicastinseln, welche miteinander nicht kommunizieren können.
Im Zuge der Future Internet Initiative wurden die nachfolgend vorgestellten Ansätze entwickelt, welche sich die Überbrückung technologischer Grenzen zum Ziel gesetzt haben. In dieser Arbeit wird ein Adapter zur Integration eines Future Internet Frameworks in die hybride Multicast-Plattform *H∀Mcast* [MCSW12] implementiert. H∀Mcast definiert einen einheitlichen Zugriff auf die Multicast-Domänen, indem es verschiedene Protokolle, auf verschiedenen logischen Ebenen des Netzwerkstacks unter einer gemeinsamen API zusammenfasst, damit Multicast-Inseln verknüpft und so Mobilität und Protokollheterogenität entgegnet. *Ariba* [HMM+10] baut lokal ein dezentrales Overlay-Netzwerk

auf, welches selbstorganisiert Ende-zu-Ende Verbindungen bei Mobiltität der Endbenutzer, Protokollheterogenität und administrativen Blockaden (z.B. NATs oder Firewalls) aufrechterhält. Ariba wird durch eine zusätzliche Ebene (*MCPO*) erweitert, welche Multicast über ein Aribanetzwerk (*SpoVNet*) implementiert.
Das Ziel von H∀Mcast ist es, einen allgemeinen, hybrigen Gruppenkommunikationsdienst zu Verfügung zu stellen. Dafür wurde eine gemeinsame Multicast-API [WSV12] entwickelt, unter der möglichst viele verschiedene Multicast-Technologien ansprechbar sind. Dies geschieht über Adapter-Module, die ein eine gemeinsame Middleware eingebunden werden. Ein Entwickler, der Multicast benutzten möchte, ist auf diese Weise nicht gezwungen für jede Multicast-Domäne eine andere API zu benutzen, bzw. in Kauf zu nehmen, dass sein Programm nur in einzelnen ausgewählten Domänen funktioniert. In dieser Arbeit wird ein Adapter implementiert, der Ariba/MCPO als ein Modul in die H∀Mcast-Middleware integriert.

2 Future Internet Technologien

Unter dem Begriff *Future Internet* sammeln sich Technologien, die sich mit der Fortentwicklung und Neustrukturierung des Internets befassen. H∀Mcast und Ariba/MCPO bieten beide die Möglichkeit per Multicast Daten zu verschicken, unterscheiden sich aber grundlegend in der Architektur.

2.1 Ariba/MCPO

Ariba ist ein Future-Internet-Ansatz, der Anwendung eine zuverlässige Ende-zu-Ende-Verbindung zwischen Netzteilnehmern zur Verfügung stellt, selbst wenn diese mobil sind, sich hinter administrativen Grenzen wie NATs oder Firewalls befinden, oder nicht direkt kommunizieren können, weil sie keine gemeinsamen Protokolle sprechen. Dabei verbirgt Ariba transparent die Heterogenität der Netze vor der Anwendung und bildet so eine zusätzliche Ebene im Netzwerkstack oberhalb der Transportschicht. Oberhalb der Ariba-Ebene befindet sich optional das Multicastmodul *MCPO*.

2.1.1 Architektur - Ariba

Die Architektur von Ariba liegt im Schichtenmodell in drei verschiedenen logischen Ebenen vor. Für Anwendungen wird eine Programmierschnittstelle in Form einer Bibliothek angeboten (Abbildung 1).

Als SpoVNet [BHMW08] wird das dezentrale, virtuelle, selbstorganisierende Netzwerk-Konzept bezeichnet, welches Ariba implementiert. Ein einzelnes SpoVNet wird als *SpoVNet-Instanz* bezeichnet und wird jeweils für einen Anwendungsfall aufgebaut. Eine SpoVNet-Instanz ermöglicht, dass auch Endpunkte kommunizieren können, die verschiedene Protokolle spechen oder zwischen denen eine Middlebox (z.B. NATs, Firewalls) liegt. Damit eine SpoVNet-Instanz funktioniert, braucht ein teilnehmender Endknoten mindestens eine funktionierende Netzwerkverbindung, über die er mit dem jeweiligen SpoVNet

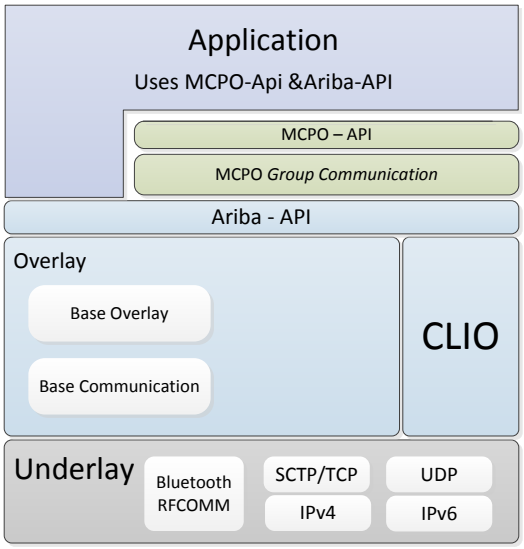

Abbildung 1: Ariba/MCPO - Architektur (nach ariba-underlay.org)

kommuniziert. Gibt es mehr als eine Internetverbindung, wird für jede der entsprechenden Underlays ein Endpunkt eingerichtet. So ist es möglich, das ein solcher Knoten zwischen Protokollen und Technologien übersetzt.

Die *Base Communication*-Schicht realisiert Verbindungen über die Protokolle im Underlay. Für die Datenübertragung innerhalb eines SpovNets wird sowohl verbindungslose Kommunikation in Form von UDP, sowie verbindungsorientierte Kommunikation mittels SCTP bzw TCP bereitgestellt. Wenn möglich wird immer IPsec oder TLS verwendet um zusätzliche Sicherheit zu bieten.
Weiterhin abstrahiert die Base Communication für darüberliegende Schichten von den Adressierungsschemata der Underlays. Jeder Endpunkt wird durch ein oder mehrere Locator, nachfolgend *Adressen* genannt, beschrieben, aber durch genau eine ID im Netzwerk identifiziert. Mittels diesem ID/Locator-Split wird Multi-Homing und Mobilität der Endgeräte ermöglicht.
Um zu überprüfen, ob eine Adresse noch aktiv ist, werden regelmäßig heart-beat-Messages mittels UDP versandt. Um Overhead zu vermeiden wird das Überprüfen der SCTP bzw. TCP - Verbindungen nur vorgenommen, wenn die Applikation die Funktion explizit aufruft.
Insgesamt werden zwischen zwei Arten von Verbindungen unterschieden:

Direkte Verbindungen liegen vor, wenn zwei Knoten nativ miteinander kommunizieren können, weil sie die gleichen Protokolle sprechen und nicht durch NATs oder Firewalls getrennt sind.

Indirekte Verbindungen liegen vor, wenn keine direkte Verbindung zwischen zwei Knoten möglich ist. In diesem Fall werden deren Daten durch einen dritten Knoten (*Relay*) überbracht.
Relays sind ausgewählte Knoten, welche zur Überbrückung von Middleboxes und Protokolldifferenzen dienen. Dies geschieht transparent mittels *NAT Relays* bzw. *Protocol Relays* (ähnlich zu *TURN*, vgl. [MMR10]). Um den Ende-zu-Ende-Charakter der Verbindungen zu bewahren, werden die Packete per Package Encapsulation und mittels Tunnel übertragen.

Die Base Communication - Schicht etabliert Multi-Homing, indem jeder Endpunkt durch den *Endpunkt Deskriptor* beschrieben wird, der eine Menge von Adressen bereitstellt unter denen der jeweilige Knoten erreichbar ist *(ID/Locator-Split)*. Ein Knoten hält dann beispielweise eine IPv4-Adresse aus dem lokalen WLAN, eine Bluetooth-RFCOMM-Adresse, sowie eine IPv4-Internetadresse über LAN. Diese Redundanz verhindert, dass bei Wegfall einer Verbindung der Knoten nicht mehr erreichbar ist. In diesem Fall wird der sendende Knoten eine andere Adresse aus dem Endpoint Descriptor des empfangenden Knotens benutzen. Diese Adresse kann dann wieder sowohl über eine direkte oder eine indirekte Verbindung erreichbar sein.
Der ID/Locator-Split ermöglicht Mobiltität. Der mobile Knoten kann den neuen Locator zur Verfügung stellen, bevor der alte Locator ungültig wird. Ob ein solcher Handover möglich ist, ist jedoch anwendungsabhängig, sodass er auf Anwendungsebene mithilfe von CLIO realisiert werden muss. Ist ein Handover nicht möglich, testet der sendende Knoten die anderen Adressen aus dem Endpunkt Deskriptor des mobilen Knotens, sobald eine Adresse ungültig geworden ist.

Das Base Overlay arbeitet mit den Verbindungen die über die Base Communication-Schicht bereitgestellt werden und kümmert sich nicht um deren Management oder um die Relays. Im SpoVNet werden die Knoten über den *Node Identifikator* adressiert. Die Hauptaufgaben des Base Overlays bestehen darin einen Node Identifikator in den zugehörigen Endpunkt Deskriptor umzuwandeln, die Knoten innerhalb einer SpoVNet Instanz zu organisieren und es stellt grundlegende DHT-Operationen zur Verfügung. Das Base Overlay benutzt Chord [SMLN$^+$03] als schlüsselbasierten Routing-Algorithmus um andere Knoten innerhalb des SpoVNets zu finden. Er versendet eine *descriptor query message*. Der gesuchte Knoten antwortet mit einer *descriptor reply message*, welche dessen Endpunkt-Deskriptor enthält. Mittels dieser Information kann ein Transportverbindung aufgebaut werden, welche Daten über die zuverlässigen (TCP/SCTP) Verbindungen der Base Communication-Schicht schickt.
Sicherheit im Base Overlay wird mittels Credentials während der Join-Phase sichergestellt. Der beitretende Knoten sendet eine *discovery message* an eine bestimmte Multicast- oder Anycastadresse. Sind diese SpoVNets verschlüsselt oder versteckt, müssen bestimmte Credentials mit angegeben werden damit die entsprechenden Bootstrap-Nodes antworten und der Join-Prozess fortgesetzt werden kann. Auf diese Weise kann der Zugang zu einem SpoVNet beschränkt und verschlüsselt werden.

Das Cross Layer Information Overlay (Clio) stellt optional Informationen der Base Communication und des Base Overlays der Anwendung zur Verfügung, falls diese auf Anwendungsebene benötigt werden.

2.1.2 Architektur - MCPO

Das MCPO-Modul realisiert einen Gruppenkommunikationsdienst auf Basis der Programmierschnittstelle von Ariba.
Als Verteilungsalgorithmus wird das *NICE-Protokoll* [BBK02] benutzt. Dieses Protokoll baut für jede Gruppe eine Clusterhierarchie über verschiedene Layer auf (siehe Abbildung 2). Jeder Knoten befindet sich in genau einem Cluster auf der untersten Ebene 0. In jedem Cluster eines Layers n, gibt es einen Cluster Leader, der sich ebenfalls in Layer n+1 befindet. Die Anzahl der Layer, ergibt sich aus den gewählten Clustergrößen. Wenn ein Knoten Daten sendet, so flutet (flooding) er diese in seinen Clustern (in mehreren, wenn der Knoten Cluster- Leader ist, ansonsten nur in *seinem* Layer 0). Die jeweiligen Cluster Leader senden die Daten an alle Mitglieder ihrer Cluster weiter, außer dem Knoten, von dem sie diese Nachricht erhalten haben. Cluster-Leader senden keine Daten in Layer 0 weiter, wenn sich der Sender im selben Cluster auf Layer 0 befindet, denn dort hat der Sender diese Daten selbst verteilt.
Nach diesem Prinzip wird jede Nachricht an alle Gruppenmitglieder verteilt. Dieses Vorgehen verringert die Last des Senders, der jetzt nur noch in *seinen* Clustern senden muss. In MCPO wird genau ein NICE-Baum aufgebaut. Das Gruppenkonzept beschränkt sich auf Filterung am Endknoten[1]. So werden alle Nachrichten im kompletten Baum geflutet, und die Knoten entscheiden anhand der „Gruppen“, denen sie beigetreten sind, welche Nachrichten sie an die überliegende Anwendung weiterreichen.

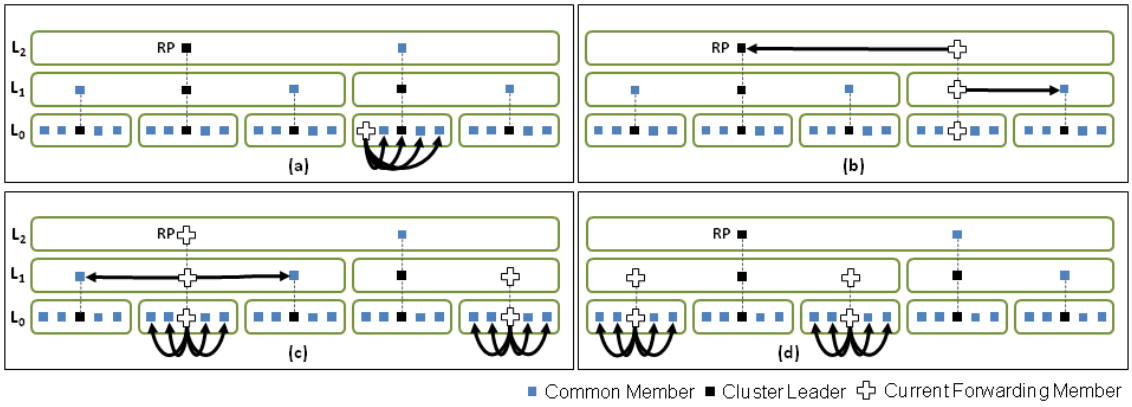

Abbildung 2: NICE-Protokoll Topologie in MCPO [2]

[1] NICE sieht konzeptionell einen separaten Baum je Gruppe vor. Dies würde eine MCPO-Instanz pro Gruppe bedeuten

[2] http://ariba-underlay.org/wiki/Documentation/MCPO

2.2 H∀Mcast

H∀Mcast ist eine hybride Multicastplattform, welche verschiedene Multicasttechnologien unter einer gemeinsamen API [WSV12] vereint. Dieses wird mithilfe von Modulen realisiert, die ein gemeinsames Interface implementieren. Diese Module können direkt Multicastprotokolle implementieren, wie z.B. Overlayprotokolle, oder sie dienen als Adapter für andere Frameworks wie z.B. MCPO/Ariba.

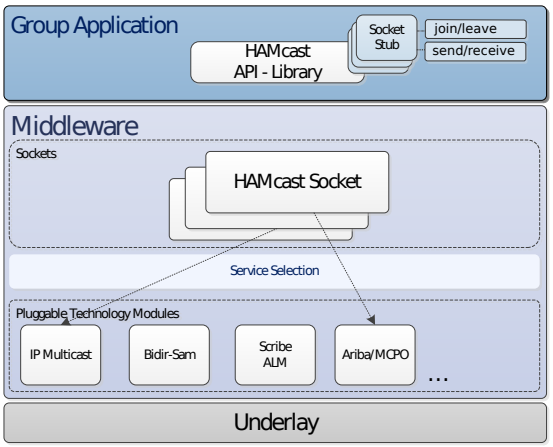

Abbildung 3: H∀Mcast-Architektur

2.2.1 Architektur

Das H∀Mcast-Konzept sieht eine universelle Gruppenkommunikationsschicht zwischen der Anwendungsebene und den verschiedenen Netzwerkstackebenen vor. Diese Schicht ist in Form einer Middleware umgesetzt, welche für die überliegenden Anwendungen eine Programmierschnittstelle bietet, mit der die unterliegenden Multicastmodule immer auf die gleiche Weise benutzen werden können. Somit ist der Anwendungsprogrammierer nicht darauf angewiesen, die benötigten Protokolle selbst zu implementieren, da das Mapping die Middleware für ihn übernimmt. Die H∀Mcast-API ist in Form von Bibliotheken für C++ und Java umgesetzt, welche die Kommunikation zwischen Anwendung und Middleware übernehmen. (Abbildung 3).
Um zwischen den Multicastdomänen zu vermitteln ist es notwendig, dass bestimmte Knoten in mehreren Domänen aktiv sind und die Pakete weiterleiten, deren Gruppen Listener in der jeweils anderen Domäne haben. Diese Aufgabe übernehmen die *Interdomain Multicast Gateways* (IMGs)(Abbildung 4).
Da in verschieden Multicastnetzen verschiedene Arten der Adressierung vorliegen, müssen die IMGs zwischen diesen abbilden.

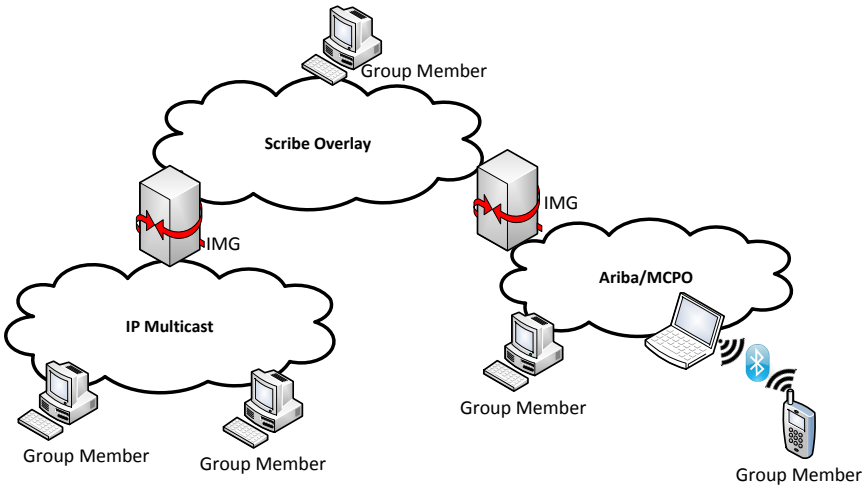

Abbildung 4: H∀Mcast-Netzwerk-Topologie

2.2.2 Adressierung

Die Adressierung in H∀Mcast folgt ähnlich wie bei Ariba dem ID/Locator-Split.
Um in H∀Mcast Gruppen voneinander zu unterscheiden wird eine global eindeutige Gruppenadressierung [WSV12] verwendet, während die Module ihre Adressierungen der jeweiligen Technologien nutzen. Diese Multicast-URI fungiert als Identifikator einer Gruppe und hat die Form:

```
scheme :// group @ instantiation : port / credentials
```

Die *Scheme* beschreibt den Namensraum (z.B. *ip* für ip-Multicast), die *Group* beschreibt die Multicastgruppe innerhalb der Scheme, die *Instantiation* eine bestimmte Quelle, z.B. bei Source-Specific-Multicast, der *Port* eine Anwendung und mittels *Credentials* können optional Authentifizierungen durchgeführt werden.

3 Umsetzung

Die H∀Mcast-Middleware soll um ein weiteres Modul erweitert werden, welches als Adapter für die Ariba/MCPO - Plattform dient (vgl. Abbildung 3) und so unter der gemeinsamen API angesprochen werde kann. Das MCPO-Modul erweitert H∀Mcast um das Multicast Overlay Protokoll NICE und mittels Ariba können Daten auf allen Transportwegen, die Ariba zur Verfügung stellt, übertragen werden. So ist H∀Mcast mithilfe von Ariba in der Lage, technologische Grenzen zu überwinden die bisher nicht im Fokus von H∀Mcast lagen, z.B. das überwinden von Firewalls, NATs und das Benutzen anderer physicher Netzarten, wie Bluetooth.

Die H∀Mcast- C++ - Schnittstelle `hamcast_module.h` wird von allen Modulen der H∀Mcast-Middleware implementiert und setzt darin jeweils die Funktionen um, welche

die Middleware aufruft. Weiterhin stellt das Modul der Middleware mithilfe von Callback Funktionen Informationen bereit, welche durch den Stack nach oben gereicht werden (z.B. wenn eine Nachricht empfangen wurde).

3.1 Adapter Architektur

Das Modul ist nach dem Adapter-Pattern implementiert und besteht aus den zwei Hauptkomponenten `mcpo_module` und `mcpo_connection` (Abbildung 5).

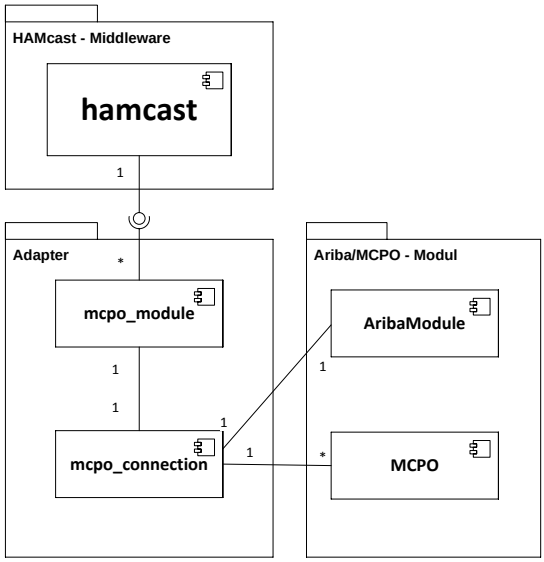

Abbildung 5: Adapter Architektur zwischen H∀Mcast und Ariba/MCPO

Die mcpo_connection ist für die Anbindung des Adapters an das MCPO-Developer-Interface zuständig. Diese Schnittstelle besteht aus mehreren zu implementierenden Interfaces aus dem MCPO/Ariba-Framework. MCPO stellt ein Interface `Receiver.h` zur Verfügung, dessen Methoden implementiert werden, um auf eingehende Nachrichten reagieren zu können (z.B. `receive_data()`). Die restlichen zu implementierenden Interfaces (`startupInterface, NodeListener`, und `Timer`) enthalten Ariba-spezifische Funktionen.
Weiterhin verwaltet die `mcpo_connection`-Klasse eine Map, welche einen H∀Mcast-URI Host auf ein MCPO-Objekt abbildet, um so eine effiziente, H∀Mcast-kompatible Adressierung umzusetzen. Dies ist notwendig, weil MCPO ein anderes Adressierungsschema für Gruppen besitzt als H∀Mcast. Es können mehreren SpoV-Nets beigetreten werden, indem Konfigurationsdaten für mehrere SpoVNets in der Middleware-Initialisierungsdatei hinterlegt werden. So stellt eine mcpo_connection mit jeweils einem Ariba-Objekt ein logisches Netzwerk-Interface dar, wie sonst z.B.

eth0. Dieses Interface ist die Verbindung zu dem unterliegenden SpoVNet welches sich durch einen kanonischen Namen identifiziert.

Das MCPO-Modul ist für die Implementation der Funktionen des `hamcast_modul`-Interfaces zuständig. Die `mcpo_connection` wird vom MCPO-Modul (`mcpo_module`, Abbildung 5) bei dem ersten *Join* erstellt, sodass keine unnötige Last für Endgerät und Netzwerk erzeugt wird und um eine logische Trennung der Zuständigkeiten zu erhalten.
Das `mcpo_module` erstellt und verwaltet die `mcpo_connection` - Instanzen. Dafür sind Anpassungen der Initialisierungsdaten nötig, da Ariba ein anderes Konfigurationsformat versteht als H∀Mcast. Das `mcpo_module` übernimmt den Umbau der Zeichenketten.

3.2 Abbildung der H∀Mcast-URI

Das oben beschriebene NICE-Protokoll erfordert prinzipiell für jede Gruppe eine NICE-Baum-Hierarchie. Ein MCPO-Objekt baut jedoch nur einen NICE-Baum auf, und benutzt ihre Gruppen ähnlich wie Ports, indem es nur auf an Endknoten filtert (vgl. 2.1.2). Um ein Baum pro Gruppe umzusetzen, wird jede H∀Mcast-URI-Group auf ein MCPO-Objekt abgebildet, sodass bei jeder neuen Gruppe ein neuer Baum erstellt wird und nicht alle Nachrichten verschiedener Gruppen im kompletten Baum geflutet werden.
In der H∀Mcast-URI wird zusätzlich zu der *Group* auch ein *Port* zur Verfügung gestellt. Dieser Port wird auf die *Gruppen* von MCPO abgebildet, welche nur am Endknoten filtern. So wird das NICE-Protokoll-Prinzip (ein Baum pro Gruppe) erhalten und zusätzlich die Gruppen von MCPO als Portkonzept verwendet.

3.3 Tests

Getestet wurde das MCPO-Adapter-Modul auf Paketdurchsatz und CPU-Belastung im Vergleich mit Ariba, sowie im Vergleich zu Ariba zusammen mit MCPO. Es handelt sich um drei identische Testprogramme, welche so viele Pakete wie möglich mit der jeweiligen Technik (Ariba, Ariba/MCPO oder H∀Mcast-Middleware + Modul) an eine Gruppe versenden. Die Empfängerseite tritt der Gruppe bei und prüft ob die Daten in der richtigen Reihenfolge ankommen und ob Pakete verloren gegangen sind. Mit einem Bash-Skript wird sekündlich die CPU-Belastung gemessen. Geschrieben wurden die Durchsatztests in C++ und auf zwei identischen Rechnern (CPU: Intel Core i7-870 8M Cache, 2.93 GH, Speicher: 8GB, Betriebsystem: Ubuntu 12.04, Ariba v0.7.1b, MCPO v0.5.1, H∀Mcast v0.5) ausgeführt, welche direkt über ein Gigabit-Netzwerk verbunden wurden.

3.3.1 Durchsatz

Der Durchsatz bestimmt sich durch Pakete, die pro Sekunde versendet werden, in Abhängigkeit von der Paketgröße (Abbildung 6). Ariba versendet wesentlich mehr Pakete pro Sekunde als Ariba/MCPO, da dieses zusätzlich das Overlay-Multicast-Netz

verwaltet, sowie einen weiteren Header vor den Ariba-Header einfügt. Als weitere Abstraktionsschicht verringert das MCPO-Adapter-Modul von H∀Mcast zusammen mit der Middleware ebenfalls den Durchsatz, da in der H∀Mcast-Middleware eine weitere Kopieroperation durchgeführt wird. Der Durchsatz für das Modul ist durch die maximale Geschwindigkeit von Ariba/MCPO begrenzt. Im Vergleich zu einem H∀Mcast Modul, welches ein Scribe [CDKR02] Overlay benutzt (bis zu 250.000 Pakete/s), oder einem Modul welches ein BIDIR-SAM-Overlay [WSW09] benutzt (bis zu 290.000 Pakete/s), ist Ariba/MCPO deutlich langsamer (siehe auch [MCSW12]). Diese Differenz erklärt sich durch den Einsatz Aribas des TCP- bzw. SCTP-Protokolls, um eine zuverlässige Verbindung zur Verfügung zu stellen. Die meisten Multicast-Anwendungen (wie auch Scribe) verwenden verlustbehaftete Protokolle und sind dementsprechend schneller, da keine gesicherte Übertragung stattfinden muss. Weiterhin ist Aribas SpoVNet ein Overlay Netzwerk, auf das MCPO ein weiteres Multicast Overlay aufbaut. Diese vielfache Overlayverwaltung verlangsamt den Versende- und Empfangsprozess.

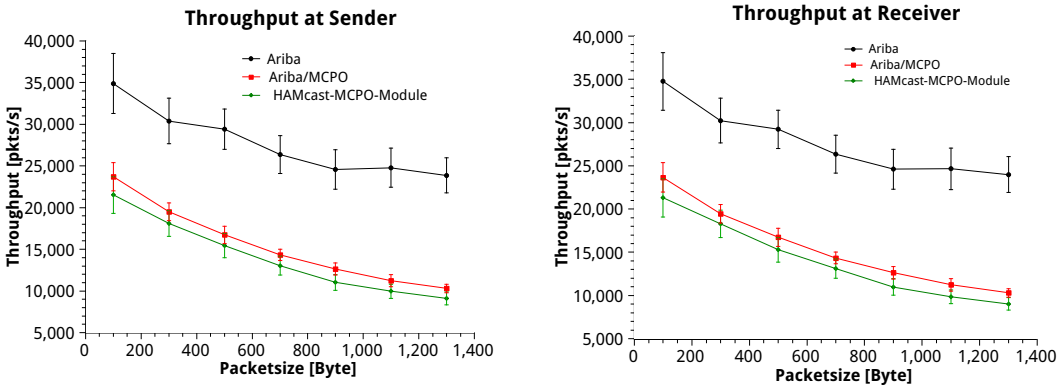

Abbildung 6: Durchsatz in Abhängigkeit von der Paketgröße

3.3.2 Paketverlust

Obwohl Ariba als unterliegende Abstraktionsschicht für MCPO, zuverlässige Verbindungen zur Verfügung stellt, kann es zu minimalen Paketverlust in Ariba/MCPO und dem Ariba/MCPO-Modul kommen. Dieser Paketverlust ist sehr gering, beläuft sich in einem Interval von einer Sekunde durschnittlich auf ein Paket und tritt sehr unregelmäßig auf (im Mittel alle 19,2 Sekunden).
In einem Testaufbau, der über die Middelware und das Modul schnellstmöglich genau 100.000 Nachrichten (200 Byte groß) versendet, wurden, an verschiedenen Stellen des entstandenen Netzwerkstacks, Zähler implementiert, um die Zuverlässigkeit der Komponenten zu überprüfen und zu bestimmen an welcher Stelle die Pakete verloren gehen. So verschwinden von dem Modul versendeten 100.000 Nachrichten, im Schnitt 2 pro Messung. Die Durchsatztest von Ariba ohne MCPO zeigen, dass Ariba eine zuverlässige Nachrichtenübertragung bietet (welches im Ariba-Konzept so vorgesehen ist), hierbei ist kein Paket verloren gegangen.

Das sendende Ariba/MCPO-Modul in der Middleware hat ebenfalls keinen Paketverlust in der zwischen Sendeapplikation und dem MCPO-Modul feststellen können.
Empfängerseitig wurde im Ariba/MCPO-Modul der Middleware jedoch der beschriebene Paketverlust festgestellt. Daraus lässt sich schließen, dass die MCPO-Komponente diese Nachrichten verliert und damit keine zuverlässige Verbindung bietet. Da der Paketverlust insgesamt sehr gering ist, wirkt sich dieses Verhalten kaum auf die Durchsatz-Messungen aus und die Messungen der Nachrichten sind bei Sender und Receiver annähernd gleich (vgl. Abbildung 6).

3.3.3 CPU-Auslastung

Die Beobachtung der Endknoten zeigt die CPU-Auslastung, welche die Sende- und Empfangsprozesse maximal bei vollem Durchsatz der Testprogramme auf den Testrechnern erzeugen (Abbildung 7).

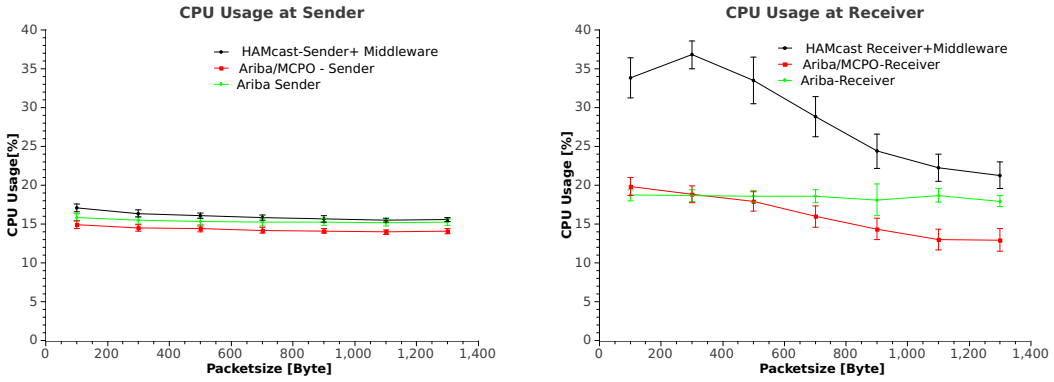

Abbildung 7: CPU-Auslastung der Testknoten bei maximalem Durchsatz

Hier wird deutlich, dass alle Sender zu ähnlicher CPU-Belastung führen, egal welche Paketgröße der Sender mit welchem Test sendet. Trotz Overhead benötigt der MCPO-Test weniger CPU-Zeit als Ariba, da er wesentlich weniger Pakete versendet (vgl. Abbildung 6).
Am Empfängerknoten sorgt die sinkende Anzahl der Pakete/s bei steigender Paketgröße für eine Verminderung der CPU-Last, da weniger Pakete kopiert werden müssen. Dieser Zusammenhang gilt für Ariba/MCPO sowie HAMcast mit Ariba/MCPO-Modul. Der Aufwand der reinen Ariba-Knoten unterscheidet sich nicht zwischen den verschiedenen Paketgrößen. Der Aufwand des Ariba-Empfängers ist demnach nicht in dem Maße von der Paketanzahl abhängig wie MCPO, sondern mehrheitlich von der absoluten Datenmenge.

Beim Empfänger ist die CPU-Auslastung des MCPO-Moduls zusammen mit der H∀Mcast-Middleware bei kleinen Paketen sehr hoch (vgl akkumulierter Graph in Abbildung 8). Grund für die hohe CPU-Auslastung ist einerseits die Middleware, welche eine Kopieroperation zusätzlich zur Technologie darunter (in disem Fall Ariba/MCPO)

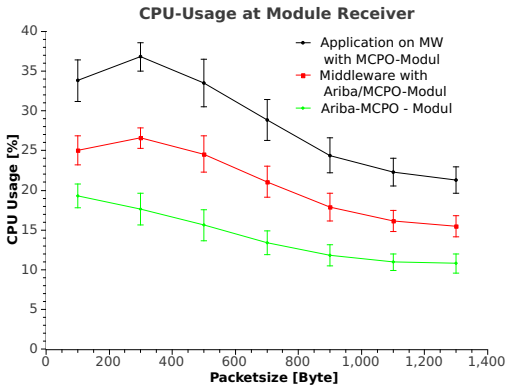

Abbildung 8: akkumulierte CPU-Last der Komponenten einer Ariba/MCPO-H∀Mcast-Anwendung

pro Paket benötigt, um die Pakete zuverlässig an die überliegende Applikation weiterzuleiten. Die Applikation benötigt weiterhin etwas CPU-Zeit um die Pakete über IPC zu empfangen. Je mehr Pakete empfangen werden, desto mehr Kopieroperation werden benötigt. Dies erklärt die erhöhte CPU-Belastung bei kleineren Paketgrößen.

4 Zusammenfassung

In dieser Arbeit wurde im Rahmen des H∀Mcast-Projektes ein Modul implementiert, um die Services des Future Internet Frameworks Ariba/MCPO in H∀Mcast zu integrieren. Dieses Modul bildet als Adapter die Schnittstelle zwischen der H∀Mcast-Middleware und Ariba/MCPO, indem es die jeweiligen APIs beider Libraries implementiert und zwischen diesen vermittelt. In H∀Mcast dient die vom Adapter implementierte Modulschnittstelle dazu, die Einbindung diverser Module in die H∀Mcast-Middleware zu ermöglichen. Jedes Modul repräsentiert dabei den Zugang zu einer Multicasttechnologie (z.B. IP-Multicast, Ariba/MCPO, Scribe). Die API von Ariba/MCPO, welche vom Adapter verwendet wird, ist die generelle Schnittstelle für Anwendungen, welche Ariba/MCPO bereitstellt. Der Adapter konvertiert die Nachrichtenformate, bildet die Adressierungen aufeinander ab und führt intitiale Funktionen aus, welche von den Plattformen benötigt werden. Dabei wurde er möglichst schlank gehalten um die Performance nicht zu beeintrachtigen.

Die Durchsatztests zeigen den hohen Kostenaufwand, den eine zuverlässige Verbindung von Ariba erfordert. Die Kosten von Ariba/MCPO sind im Vergleich zu Ariba, um ein Drittel erhöht und gehen größtenteils auf die Verwaltung des NICE-Overlays zurück. Paketverlust trat bei Ariba/MCPO auf (etwa 2 Pakete alle 19 Sekunden), nicht aber bei Ariba ohne MCPO. Die CPU-Belastung zeigt nur beim Empfänger deutliche Unterschiede zwischen Ariba und Ariba/MCPO. Die CPU-Belastung des Ariba/MCPO-Empfängers ist im wesentlichen von der Paketanzahl, nicht aber vom gesamten Datendurchsatz abhängig. Dadurch sinkt die CPU-Last beim Ariba/MCPO-Empfänger bei steigender Paketgröße (der Sender schickt dann deutlich weniger Pakete)

Das Adaptermodul produziert zusammen mit der H∀Mcast-Middleware im Vergleich zu Ariba und Ariba/MCPO nur wenig mehr CPU-Last beim Senden aber wesentlich mehr CPU-Last beim Empfangen der Pakete, je mehr Pakete empfangen werden. Auch hier führen weniger, dafür größere Pakete, zu einer Verringerung der CPU-Last.
Letztendlich bietet das H∀Mcast mit Ariba/MCPO-Modul ein stabiles Overlay mit minimalem Paketverlust, welches Middleboxes umgehen und Endgeräte über diverse physische Netzarten ansprechen kann.

5 Ausblick

Die Performance von Ariba/MCPO ist im Vergleich zu anderen Multicastprotokollen gering. Der Durchsatz des Adaptermoduls grenzt an diese Durchsatzrate. Soll hier eine Verbesserung stattfinden, muss zuerst Ariba/MCPO verbessert werden, da dessen Durchsatz die Obergrenze für das Adaptermodul bildet. Es ist abzusehen, dass Ariba in Zukunft in dieser Hinsicht verbessert wird.
Multicast hat nicht den Anspruch, zuverlässig zu sein. Zuverlässigkeit und hohe Stabilität ist aber eines der Hauptziele von Ariba. MCPO garantiert keine zuverlässige Verbindung, da Zuverlässigkeit bei Multicast nicht üblich ist. Dadurch kommt es zu minimalem Paketverlust, obwohl Ariba, als unterliegende Schicht, zuverlässig ist. Es stellt sich die Frage ob bei einer Kombination von Ariba und MCPO es mehr Sinn ergeben würde diesen Multicast wirklich "zuverlässig" zu gestalten.
Die Tests haben gezeigt, dass es nicht ausreichend ist, auf eine darunterliegende, zuverlässige Netzwerkkommunikation zu vertrauen. So wäre es untersuchenswert, wie man Zuverlässigkeit, im Bezug auf Multicastprotokolle und Implementationen auf höherliegenden Abstraktionsschichten, erreichen kann.
Das Gruppenkonzept, welches MCPO umsetzt, baut genau einen Baum auf, über den die Daten aller Gruppen versendet werden. Für große Gruppen mit vielen Sendern ist dieses Konzept wahrscheinlich nicht geeignet, da es nicht gut genug skaliert. Diese Problematik kann umgangen werden, indem mehrere MCPO-Instanzen, eine für jede Gruppe, verwendet werden. Es wäre auszuwerten, inwiefern dieser Ansatz diese Problematik verbessert.
Ariba/MCPO und H∀Mcast verwenden beide einen komplexen logischen Netzwerkstack. Es wäre zu überprüfen, welche Komplikationen durch die Kombination dieser Stacks entstehen, ob sich die Frameworks gegenseitig negativ beeinflussen und ob der Overhead, für das erreichte Ergebnis, vertretbar ist. Weiterhin wäre es untersuchen, inwieweit sich die Ziele und Konzepte von H∀Mcast und Ariba/MCPO überschneiden bzw. widersprechen und welche Folgen das mit sich bringt.

Literatur

[BBK02] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy. Scalable Application Layer Multicast. In *Proc. of SIGCOMM '02*, pages 205–217, New York, NY, USA, 2002. ACM Press.

[BHMW08] Roland Bless, Christian Hübsch, Sebastian Mies, and Oliver Waldhorst. The underlay abstraction in the spontaneous virtual networks (spovnet) architecture. In *Proc. 4th EuroNGI Conf. on Next Generation Internet Networks (NGI 2008)*, pages 115–122, Krakow, Poland, 2008.

[CDKR02] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 20(8):100–110, 2002.

[HMM+10] Christian Hübsch, Christoph P. Mayer, Sebastian Mies, Roland Bless, Oliver P. Waldhorst, and Martina Zitterbart. Reconnecting the internet with ariba: self-organizing provisioning of end-to-end connectivity in heterogeneous networks. *SIGCOMM Comput. Commun. Rev.*, 40(1):131–132, January 2010.

[MCSW12] Sebastian Meiling, Dominik Charousset, Thomas C. Schmidt, and Matthias Wählisch. HAMcast – Evaluierung einer systemzentrierten Middleware-Komponente für einen universellen Multicast-Dienst im Future Internet. *Praxis der Informationsverarbeitung und Kommunikation (PIK)*, 35(2):83–89, Mai 2012.

[MMR10] R. Mahy, P. Matthews, and J. Rosenberg. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). RFC 5766, IETF, April 2010.

[SMLN+03] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.

[WSV12] Matthias Wählisch, Thomas C. Schmidt, and Stig Venaas. A Common API for Transparent Hybrid Multicast. IRTF Internet Draft – work in progress 06, IRTF, August 2012.

[WSW09] Matthias Wählisch, Thomas C. Schmidt, and Georg Wittenburg. BIDIR-SAM: Large-Scale Content Distribution in Structured Overlay Networks. In Mohamed Younis and Chun Tung Chou, editors, *Proc. of the 34th IEEE Conference on Local Computer Networks (LCN)*, pages 372–375, Piscataway, NJ, USA, October 2009. IEEE Press.