HΛMcast – Evaluierung einer systemzentrierten Middleware-Komponente für einen universellen Multicast-Dienst im Future Internet

Sebastian Meiling

Dept. Informatik
Hochschule für Angewandte
Wissenschaften Hamburg
Berliner Tor 7
20099 Hamburg
Deutschland
sebastian.meiling@haw-hamburg.de

Sebastian Meiling studierte Informatik an der Universität Leipzig, seine Diplomarbeit schrieb er zum Thema TCP with Adaptive Pacing für drahtlose Mesh-Netze. Seit 2009 ist er wissenschaftlicher Mitarbeiter in der Arbeitsgruppe Internet Technologies (iNET) an der Hochschule für Angewandte Wissenschaften Hamburg (HAW). Seine Forschungsschwerpunkte liegen im Bereich zukünftiger Internet Technologien und Protokolle, mit Fokus auf Gruppenkommunikation und effizienter Datenverteilung.

Dominik Charousset

Dept. Informatik
Hochschule für Angewandte
Wissenschaften Hamburg
Berliner Tor 7
20099 Hamburg
Deutschland
dominik.charousset@haw-hamburg.de

Dominik Charousset studiert Informatik an der Hochschule für Angewandte Wissenschaften Hamburg (HAW) im Masterstudiengang. Seit 2009 ist er dort wissenschaftlicher Mitarbeiter in der Arbeitsgruppe Internet Technologies (iNET). Seine Forschungsschwerpunkte liegen im Bereich der Programmierung verteilter und nebenläufiger Systeme, insbesondere auf Basis des Aktormodells.

Thomas C. Schmidt

Dept. Informatik
Hochschule für Angewandte
Wissenschaften Hamburg
Berliner Tor 7
20099 Hamburg
Deutschland
t.schmidt@ieee.org

Prof. Dr. *Thomas C. Schmidt* leitet die Arbeitsgruppe Internet Technologies (iNET) an der Hochschule für Angewandte Wissenschaften (HAW) Hamburg. Er studierte Mathematik, Physik und Germanistik an der Freien Universität Berlin, der University of Maryland und der Technischen Universität Berlin. 1993 promovierte er zur Quantendynamik von Vielteilchensystemen bei Prof. Dr. D. H. E. Groß. Seine Forschungsschwerpunkte umfassen derzeit das Internet der nächsten Generation, mobile und multimediale Kommunikation, P2P-Netze sowie Hypermedia-Systeme. Thomas Schmidt beteiligt sich aktiv an der Internet-Standardisierung.

Matthias Wählisch

Freie Universität Berlin
Inst. für Informatik
Takustr. 9
14195 Berlin
Deutschland
waehlisch@ieee.org

Matthias Wählisch studierte Informatik und Neuere deutsche Literatur an der Freien Universität Berlin, an der er zum Thema hybrides, strukturiertes Multicast Routing diplomierte. Seit 2009 setzt er dort seine Arbeit als wissenschaftlicher Mitarbeiter in der Arbeitsgruppe Technische Informatik und Telematik fort. Seine Forschungsschwerpunkte liegen in der effizienten Kommunikation von Internet-Teilnehmern. Dies schließt den Entwurf und die Analyse von Internet-Protokollen, mobile Datenübertragung und P2P-Netze sowie die Untersuchung der Internet-Struktur ein.

***Zusammenfassung*—Das Internet hat die globale Kommunikation durch eine einzelne, adaptive Abstraktionsschicht revolutioniert. IP verbindet heute Millionen von Anwendungen, welche wiederum über die Socket API an die bisherige IP-Schicht gebunden sind. Eine Veränderung dieser 'Verbindungsschicht' ist in der Praxis nur schwer erreichbar. In diesem Beitrag präsentieren wir den HΛMcast-Ansatz einer Multi-Service-Architektur für ein Future Internet auf höherem Abstraktionsniveau. Für das Publish/Subscribe Modell eines universellen Multicast-Dienstes integrieren wir heterogene Netzwerktechnologien unter einer transparenten Anwendungsschnittstelle mit abstraktem Namensschema und einer systemzentrierten Middleware-Komponente. Eine gründliche Evaluierung der Leistungsdaten unseres HΛMcast-Prototypen kann in diesem Beitrag zeigen, dass die entstehenden Anforderungen auf gegenwärtigen Endsystemen problemlos bewältigt werden können.**

I. Einleitung

Der Erfolg des Internets in den letzten 30 Jahren basiert auf der allgegenwärtigen Verfügbarkeit eines globalen IPv4-Unicast-Dienstes und auf dessen einfacher Benutzung über die Berkeley Socket-API. Zwischenzeitlich hat sich die Art der Kommunikation im Internet gewandelt. Viele populäre Anwendungen wie das Web basieren auf einem inhaltszentrierten Publish/Subscribe Prinzip, viele weitere wie IPTV, MMORGs, soziale Netzwerke und Konferenz-Software verteilen zusätzlich Daten an Gruppen von Teilnehmern. Für eine effiziente Gruppenkommunikation sind Multicast-Verfahren besonders geeignet. Allerdings sind Netzwerkdienste wie IP-Multicast [1] nur dann anwendungsgeeignet, wenn sie überall und transparent benutzt werden können. Dies ist gegenwärtig auch aufgrund der Multicast-Diversität (IP vs. ALM, ASM vs. SSM) nicht gewährleistet, da die Anwendungsschnittstellen untereinander nicht kompatibel sind. Programmierer müssen

daher die Multicast-Technologie bereits zur Programmierzeit kennen und festlegen.

In diesem Beitrag stellen wir die H∀Mcast Multi-Service-Architektur für ein Future Internet vor, welches neue Netzwerkdienste inkrementell verteilbar und evolutionär entwickelbar machen möchte. Wir präsentieren die Implementierung und Evaluierung des hybriden Multicast von H∀Mcast, der mittels einer transparenten Anwendungsschnittstelle nebst abstraktem Namensschema, einer Middleware-Komponente zur Technologie-Abstraktion sowie Gateways zur Überwindung administrativer und technologischer Grenzen realisiert wird.

Im Weiteren gliedert sich die Arbeit wie folgt: Grundlegende Probleme sowie verwandte Arbeiten aus dem Bereich der Gruppenkommunikation werden in Abschnitt II diskutiert. In Abschnitt III stellen wir die Architektur von H∀Mcast vor und beschreiben anschließend die Implementierung (IV) des Software-Prototypen. Die Evaluierung erläutern wir in Abschnitt V und schließen mit Zusammenfassung und Ausblick.

II. Problemstellung und verwandte Arbeiten

IP-Multicast [1] bietet einen standardisierter Netzwerkdienst zur effizienten Gruppenkommunikation, der über eine erweiterte Socket-API benutzbar ist. Allerdings blieb seine Verbreitung im Internet – im Gegensatz zu IP-Unicast – gering. Dies hat vielfältige Ursachen und ist unter anderem durch ein widerstreitendes Anreizsystem [2] zwischen ISPs sowie Inhaltsanbietern und Nutzern bedingt. ISPs bietet Multicast zwar den Vorteil einer verringerten Netzwerklast, allerdings sorgen reduzierte Datenraten für verminderte Einnahmen und die hohe Komplexität (Gruppenverwaltung) sowie Offenheit (keine Autorisierung) für erhöhte Betriebskosten. Die physische wie betriebswirtschaftliche Kostenreduktion erfolgt vor allem beim Inhaltsanbieter (Sender), der die Daten nur einmal an die Gruppe statt an jeden einzelnen Empfänger übertragen muss.

Dieses Deployment-Dilemma von IP-Multicast wurde erst kürzlich mit der Einführung von IPTV aufgelöst, indem Netzwerkbetrieb und Dienstangebot hier in einer Hand liegen. Entsprechend hat IP-Multicast in 'walled gardens' der IPTV-Anbieter in den letzten Jahren große Verbreitung gefunden. Für ISP-ungebundene Anwendungsentwickler und -betreiber bleiben diese Infrastrukturen aber verborgen, so dass die Mehrzahl der Gruppenanwendungen entweder Server-zentriert oder mittels Overlay-Multicast Client-seitig proprietäre Lösungen verwendet.

IP-Multicast besteht weiter im MBone (IPv4) und M6Bone (IPv6) aus verteilten Netzwerkinseln, welche über statische Unicast-Tunnel verbunden sind. Als Erweiterung dieses Ansatzes werden dynamische Verfahren wie das *Automatic IP Multicast without explicit Tunnels* (AMT) [3] entwickelt. Für AMT ist eine Anpassung bestehender Gruppenanwendungen nicht notwendig, aber die Netzwerk-Infrastruktur muss um AMT-Gateways und -Relays erweitert werden. AMT bleibt jedoch auf IP-Layer Multicast beschränkt und integriert IP-Versionen nicht. Overlay-Multicast-Verfahren wie CAN [4], Scribe [5] oder BIDIR-SAM [6] stellen Gruppenkommunikation mittels P2P-Techniken auf Anwendungsebene bereit. Zwar lässt sich Multicast somit problemlos oberhalb der Netzwerkschicht einsetzen, allerdings steht der Dienst i.d.R. nur einer Anwendung auf dem Endgerät zur Verfügung.

Hybride Multicast-Technologien vereinen schichtübergreifend die Vorteile von IP-Multicast (Effizienz) und Overlay-Multicast (Verfügbarkeit), um einen universellen Gruppenkommunikationsdienst bereitzustellen. Dabei werden *IP-Multicast Inseln* über ein P2P-Overlay zu einer Multicast-Domäne verbunden. Beispiele hybrider Multicast-Architekturen sind *Island Multicast* (IM) [7], *Scalable Hybrid Multicast* (SHM) [8], *Universal Multicast* (UM) [9] und *Hybrid Shared Tree* [10]. Diese Ansätze erweitern die Gruppenkommunikation und überwinden die Grenzen von IP-Multicast, doch bleiben sie teilweise bezüglich der Übertragungstechnologien unflexibel. So unterstützt UM zwar unterschiedliche Overlay-Protokolle, eine Kommunikation zwischen verschiedenen Overlays oder den IP-Versionen ist aber nicht vorgesehen.

Die Performanz hybrider Multicast-Verfahren hängt zum einen von der Unterstützung von Intra-Domain Multicast auf der Netzwerkschicht, zum anderen vom gewählten Overlay-Schema auf dem Inter-Domain-Level ab. Mittels einer a priori Abschätzung [11] konnte gezeigt werden, dass hybride Multicast-Ansätze das Potential besitzen, dem Leistungsniveau eines globalen IP-Multicast-Dienstes nahe zu kommen. Die Ende-zu-Ende-Verzögerung wird dabei maßgeblich durch den Aufbau des Unicast-Overlays und der Wahl des Overlay-Multicast-Protokolls beeinflusst.

Eine zunehmende Anzahl von Forschungsaktivitäten widmet sich dem namensorientierten Publish/Subscribe Paradigma auf der Netzwerkschicht (s. [12] als Übersicht). Dabei wird Multicast als 'alter Verwandter' erst langsam in der Diskussion wahrgenommen [13]. Der namensorientierte H∀Mcast-Ansatz wäre geeignet, ein Brücke zwischen diesen Parallelkonzepten zu schlagen.

III. Die H∀Mcast-Architektur

H∀Mcast [14] erstellt eine universelle Gruppenkommunikationsschicht mithilfe eines hybriden, technologieintegrierenden Multicast. Im Gegensatz zu anderen hybriden Ansätzen basiert H∀Mcast auf einem generischen Konzept. Statt isolierte IP-Multicast-Netzwerke über ein spezifisches Unicast-Overlay miteinander zu verbinden, existiert jede Gruppe unter ihrem Namen verteilt auf Multicast-Domänen einer bestimmten Technologie (Abb. 1). Administrative und technologische Domänengrenzen werden mittels *Interdomain Multicast Gateways* (IMGs) überwunden.

A. Middleware-Konzept

Der hybride Multicast-Dienst von H∀Mcast wird mithilfe einer abstrakten Programmierschnittstelle [15] umgesetzt und erfordert die Bereitstellung einer erweiterten Vermittlungsintelligenz auf den Endsystemen. Die zugrundeliegende Idee des hybriden Dienstes basiert auf der Beobachtung, dass viele Knoten im Internet – insbesondere Endsysteme, Edge-Router und Service-Gateways – heute über ausreichende Ressourcen (Rechenleistung, Speicher) verfügen, um auch komplexere

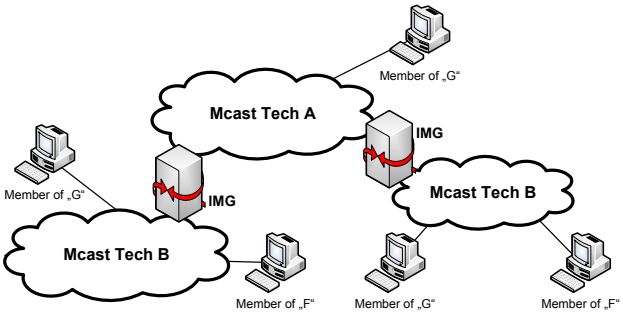


Abbildung 1. Netzwerk-Topologie mit mehreren Multicast-Domänen unterschiedlicher Technologie (A, B), die über IMGs verbunden sind. Die Gruppenmitglieder sind dabei domänenübergreifend verteilt.

Funktionen zur Paketvermittlung auszuführen. Die hybride Architektur ist dabei nur auf ein Unicast-Routing für den Overlay-Multicast angewiesen und entkoppelt so Netzwerkinnovation von Netzwerkkompetenz der Anwendungsentwickler und auch vom Kooperationswillen der Netzbetreiber.

Die erweiterte Vermittlungsintelligenz des Endsystems ist in einer systemzentrierten Middleware-Komponente angesiedelt. Diese stellt Funktionen zur Erkennung und Konfiguration lokaler Ressourcen bereit und insbesondere die zur Laufzeit verfügbaren Multicast-Technologien. Weiterhin erlaubt es die Middleware, Multicast-Technologien auszuwählen sowie Gruppendaten zwischen verschiedenen Technologien und Domänen zu vermitteln.

B. Gruppennamen und -adressen

Eine Gruppe repräsentiert eine Menge von Netzteilnehmern (Empfänger), die von den Sendern entkoppelt agieren. Traditionell sind Name und Adresse einer Gruppe jeweils abhängig von der verwendeten Multicast-Technologie. Innerhalb einer Technologie identifiziert der Name eine Multicast-Gruppe eindeutig und die Adresse wird zur Lokalisierung der Gruppenteilnehmer sowie Datenverteilung verwendet. Da die Gruppenkommunikation in HΛMcast technologieübergreifend und für Anwendungen transparent bleibt, müssen die Repräsentation von Adressen und Namen der Multicast-Gruppen getrennt werden (Loc-ID Split). Eine Multicast-Gruppe kann folglich bei HΛMcast anhand ihres Namens domänenübergreifend (global) eindeutig identifiziert werden. Die Gruppennamen werden deshalb in Form von Multicast URIs (*Uniform Resource Identifier*) kodiert [15], die wie folgt augebaut sind:

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

Das *scheme* spezifiziert den Namensraum, z.B. SIP: `sip://snoopy@peanuts.com:4321`. Die *group* identifiziert eine Multicast-Gruppe innerhalb des definierten Namensraums eindeutig. Die *instantiation* referenziert eine dedizierte Quelle oder eine Hostgruppe (z.B. Overlay), welche die angegebene Gruppe instantiiert (z.B. SSM-Gruppe $< S, G >$), und der Port die Anwendung. Mithilfe der (optionalen) *credentials* können Zugriffs- oder Inhalts-Authentifizierungen abgebildet werden. Letzteres stellt eine Erweiterung gängiger Verfahren wie IP-Multicast dar, die im Allgemeinen offen und ohne Teilnehmer-Authentifizierung sind.

Das *Mapping* von Gruppennamen zu technologiespezifischen Gruppenadressen erfolgt innerhalb einer Domäne sowie an den Grenzen zwischen zwei Domänen bei einem Gateway (IMG). Jede Multicast-Domäne besitzt ein lokal gültiges Mapping mit einer zugehörigen Mapping-Funktion. Dieses kann *stateless* auf Basis einer Heuristik oder aber *stateful* erfolgen. Beispielsweise lässt sich ein Gruppenname in einer Domäne mit Overlay-Multicast (z.B. Scribe) über die Hashfunktion des Overlays in eine Adresse (Hash) umwandeln, für IPv4-Domänen wäre aufgrund des kleinen Adressbereichs und der damit möglichen Adresskollisionen ein lokaler Verzeichnisdienst zu verwenden.

C. Interdomain Multicast Gateway (IMG)

In HΛMcast kann Multicast über verschiedene Technologien realisiert werden. Es ist daher notwendig die Daten einer Gruppe zwischen den verschiedenen Multicast-Domänen, entsprechend der Verteilung von Sendern und Empfängern, zu vermitteln. Diese Funktionalität wird von den IMGs bereitgestellt. Dabei handelt es sich um eine Erweiterung des in [10] vorgestellten Konzepts. Ein IMG hat also Zugriff auf mehrere Multicast-Domänen und kann Daten zwischen diesen auszutauschen (s. Abb. 1). Da der Gruppenname zwar global eindeutig, die Gruppenadresse jedoch an eine Multicast-Domäne gebunden ist, muss der IMG ein entsprechendes Mapping kennen.

IV. Middleware-Implementierung

HΛMcast bietet dem Programmierer einen universellen Multicast-Dienst mit einer stabilen, plattformunabhängigen API. Die Abstraktion von system- und technologieabhängigen Funktionen wurde durch die zuvor beschriebene middlewarebasierte Architektur erreicht. Durch ein *Late Binding* der Multicast-Technologie können Gruppenanwendungen unter Verwendung der HΛMcast-API entwickelt werden, ohne dass zur Kompilierzeit die zur Laufzeit benutzte Technologie weder verfügbar noch bekannt sein muss. Somit verringert sich zum einen der Entwicklungsaufwand einer Software, weil ein Quellcode alle möglichen Szenarien sowie von HΛMcast unterstützte Betriebssysteme abdeckt. Zum anderen profitieren diese Anwendungen ohne Änderung von neuen Multicast-Technologien oder einer besseren Verfügbarkeit von nativem IP-Multicast.

A. Client-Bibliothek

Die Multicast-API ist derzeit für den Prototypen als C++-Bibliothek sowie Java-Package verfügbar. Sie ist konform zum IRTF-Draft [15] implementiert und stellt sowohl alle standard Multicast-Calls als auch erweiterte Service-Calls und Socket-Options bereit. So ist es möglich den aktuellen Middleware-Status abzufragen, um z.B. eine Liste verfügbarer Multicast-Technologien oder Gruppen zu erhalten. Weiterhin existieren ereignisbasierte *Callbacks*, die es erlauben Veränderungen des Multicast-Status einer Technologie zu erkennen und darauf

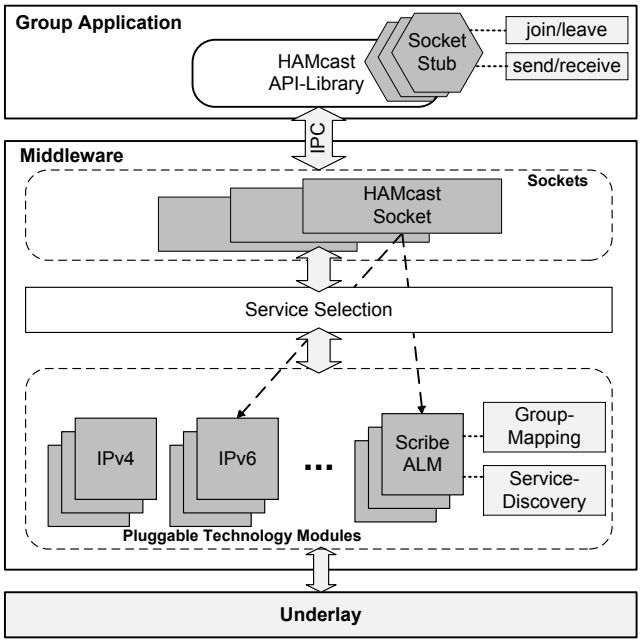


Abbildung 2. Aufbau und Komponenten des H∀Mcast-Prototypen.

zu reagieren. Diese Funktionalität ist insbesondere für die Implementierung der IMGs erforderlich.

Statt den tatsächlichen Sockets benutzen H∀Mcast-basierte Anwendungen sogenannte *Stubs* (Abb. 2), die ausschließlich Gruppennamen als Eingabeparameter akzeptieren. Alle Operationen werden zur Laufzeit über einen Kanal zur *Inter Process Communication* (IPC) an die Middleware weitergeleitet, wo die Stubs schließlich an die entsprechenden Sockets gebunden werden. Die Kommunikation zwischen Gruppenanwendung und H∀Mcast-Middleware erfolgt dabei auf Basis eines selbstentwickelten, leichtgewichtigen Protokolls. Da die IPC-Daten über *Localhost*-Sockets ausgetauscht werden, lässt sich die Multicast-API auch auf weitere Plattformen portieren.

B. Middleware-Komponente

Die Middleware abstrahiert die Gruppenkommunikation von der verwendeten Multicast-Technologie. Ihre Aufgabe besteht primär in der Verwaltung von Technologiemodulen sowie deren Auswahl und Anbindung an die Multicast-Sockets einer Anwendung. Jedes Modul implementiert eine Multicast-Technologie (z.B. IP- oder Overlay-Multicast) und kann dynamisch zur Laufzeit in die Middleware eingebunden werden. Die Kommunikation zwischen der Middleware und einem Modul wird über ein generisches C-Interface realisiert.

Beim Start führt jedes Modul seine *Service-Discovery* durch und ermittelt so, ob die bereitgestellte Multicast-Technologie in der lokalen Netzwerkumgebung anwendbar ist. Verfügt ein Endsystem über mehrere physische Netzwerkverbindungen, können ggf. mehrere Instanzen des Technologiemoduls geladen werden. Diese stehen in der Middleware anschließend als (virtuelle) Multicast-Interfaces zur Verfügung. Ergibt die Service-Discovery, dass eine Technologie nicht verfügbar ist, wird das Modul wieder entladen. Somit sind zur Laufzeit in der Middleware nur die tatsächlich verfügbaren Technologien aktiv.

V. Performanz-Evaluierung

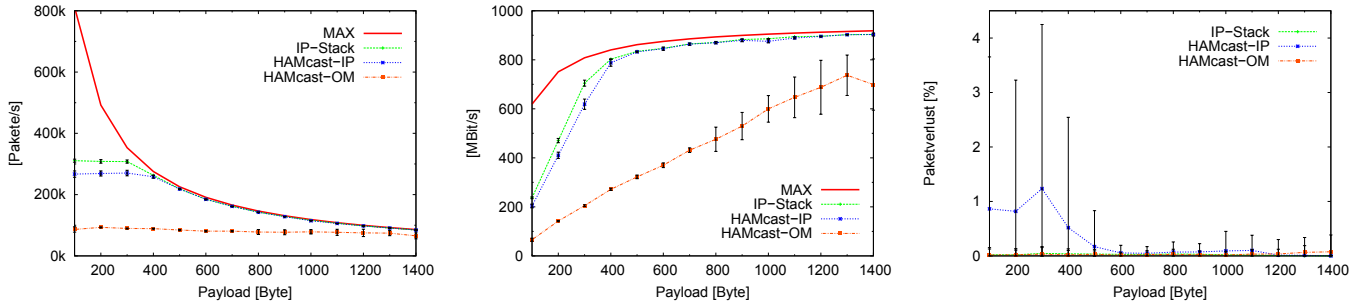
Um realistische Messdaten zur Systemperformanz und zum Overhead der H∀Mcast-Middleware zu bekommen, wurde der Prototyp mit dem IP-Stack des Linux-Kernel verglichen. Als Multicast-Technologien wurden für H∀Mcast sowohl IP-Multicast als auch Overlay-Multicast (OM) auf Basis von Scribe eingesetzt. Der Versuchsaufbau bestand aus zwei Endsystemen (Sender und Empfänger), die, zur Vermeidung externer Netzwerkeinflüsse, direkt verbunden waren. Es wurden Rechner mit einer QuadCore-CPU und 8 GB RAM verwendet, die Bandbreite der Netzwerkverbindung betrug 1 Gbit/s. Als Betriebssystem kam Ubuntu 10.10 mit Linux-Kernel der Version 2.6.35 zum Einsatz.

Pro Versuch wurden 25 Durchläufe mit einer Dauer von 60 s durchgeführt, wobei die ersten und die letzten 10 Sekunden verworfen wurden. Für die Auswertung wurden die verbleibenden 40 s bei einem Intervall von einer Sekunde verwendet, daraus ergaben sich je Versuch 1000 Messwerte. Gemessen wurden der Paket- sowie der korrespondierende Datendurchsatz, die Paketverluste und die CPU-Auslastung bei der Übertragung von Paketen mit Nutzdaten (Payload) von 100 bis 1400 Bytes (B) in 100 B Schritten. Die maximale Payload wurde in den Versuchen auf 1400 B begrenzt, da die maximale Ethernet-MTU bei 1500 B liegt. Zieht man UDP-Header (8 B) sowie IP-Header (20 B) ab, verbleiben 1472 B für Nutzdaten je Paket. Bei H∀Mcast-OM muss zudem der Overlay-Header von 90 B berücksichtigt werden.

A. Ergebnisse

In den nachfolgenden Graphen sind die Ergebnisse der Performanz-Evaluierung in Form von Durchsatz, Paketverlust und CPU-Auslastung dargestellt. Die Stützpunkte in den Graphen ergeben sich jeweils aus den Mittelwerten über 1000 Messpunkte, die Fehlerbalken geben die Standardabweichung vom Mittelwert der betrachteten Messgröße an.

1) Durchsatz: Ein wesentliches Kriterium für die Performanz war der Vergleich des Durchsatzes in `Pakete/s` sowie `Mbit/s` beim Sender und Empfänger, siehe Abbildung 3. Die Graphen zum Paketdurchsatz (Abb. 3a) sowie dem korrespondierende Datendurchsatz (Abb. 3b) beim Sender zeigen, dass der IP-Stack einen maximalen Durchsatz von etwa 320.000 `Pakete/s` erreicht, während das Maximum bei H∀Mcast-IP unter 300.000 `Pakete/s` liegt. Die Leistung des IP-Stacks wird dabei im Bereich 100 – 400 B durch die verwendete Hardware (CPU, RAM, Netzwerkkarte) begrenzt. Bei H∀Mcast schlägt sich der Overhead durch Middleware und *Inter-Prozess-Kommunikation* (IPC) nieder, so dass der Durchsatz gegenüber dem IP-Stack um ca. 40.000 `Pakete/s` geringer ist. Ab einer Payload von 500 B liegen IP-Stack und H∀Mcast-IP gleich auf, wobei nun die Netzwerkbandbreite (`MAX`) der begrenzende Faktor ist. Der Paketdurchsatz von H∀Mcast-OM (Scribe) liegt über den gesamten Bereich bei ca. 90.000 `Pakete/s`, er wird wesentlich durch das P2P-Overlay-Netzwerk beeinflusst. Das Absinken des Datendurchsatzes bei 1400 B ist durch Überschreiten der MTU und



(a) Paketdurchsatz (Sender) (b) Datendurchsatz (Sender) (c) Paketverluste (Empfänger)

Abbildung 3. Paket- und Datendurchsatz sowie Paketverluste für IP-Stack, H∀Mcast-IP und H∀Mcast-OM .

der damit verbundenen Paket-Fragmentierung begründet. Der Durchsatz auf Empfängerseite verhält sich ähnlich wie bei dem Sender, was die Analyse der Paketverluste bestätigt.

2) Paketverlust: Abbildung 3c zeigt die Ergebnisse zur Evaluierung der Paketverluste auf Empfängerseite. Bei geringer Payload ($<$ 500 B) weist H∀Mcast-IP eine niedrige Verlustwahrscheinlichkeit auf, welche im Mittel nicht über 1,2 % hinausging. Die Paketverluste bei H∀Mcast-OM liegen für alle Payloadgrößen, ähnlich dem IP-Stack, bei nahe 0 %, allerdings ist der Datendurchsatz beim Overlay-Multicast im Vergleich zum IP-Stack sowie H∀Mcast-IP geringer. Die teils deutlichen Unterschiede in der Standardabweichung bei H∀Mcast ergeben sich aufgrund von Empfangspuffern in der Middleware, die die Paketverluste zunächst auffangen, letztlich jedoch nicht verhindern können. In der Summe liegen die Paketverluste ab 500 B aber gleich auf.

3) CPU-Auslastung: Die Ergebnisse zur CPU-Auslastung sind in Abbildung 4 dargestellt und entsprechend der Anteile für Anwendung, Middleware und Kernel aufgeteilt. Auf der Y-Achse ist die Auslastung entsprechend der vier CPU-Kerne von 0 bis 400 % angegeben. Die X-Achse ist wie zuvor bezüglich der Nutzdaten je Paket in Bytes eingeteilt.

Insgesamt liegt die Prozessorlast bei H∀Mcast über dem Linux IP-Stack. Während auf Senderseite der IP-Stack inklusive Anwendung insgesamt etwa einen CPU-Kern (100 %) voll auslasten würde, liegt H∀Mcast-OM bei ca. 200 % und H∀Mcast-IP teilweise bei über 300 %. Auf Empfängerseite reduziert sich die CPU-Auslastung in allen Fällen und liegt für H∀Mcast (IP + OM) über den gesamten Bereich bei ca. 150 % sowie beim Linux IP-Stack bei ca. 50 %. Betrachtet man die Verteilung der Auslastung auf die einzelnen Komponenten, zeigt sich, dass insbesondere der Kernel-Anteil bei H∀Mcast deutlich höher ausfällt. Grund dafür ist die Inter-Prozess-Kommunikation zwischen Middleware und Client-Anwendung, welche bei unseren Softwareprototypen auf Basis von *Localhost-Sockets* und einem eigenen IPC-Protokoll (siehe IV) realisiert wurde. Zu beachten ist dabei aber auch, dass neben der Übersetzung von Gruppennamen zu -adresse (*Mapping*) gegenüber standard IP-Multicast zwei zusätzliche Kopieroperationen (Anwendung/IPC + IPC/Middleware) notwendig sind. Damit erhöht sich der Kernel-Anteil an der CPU-Auslastung, wobei der Anteil der Anwendung annähernd gleich bleibt.

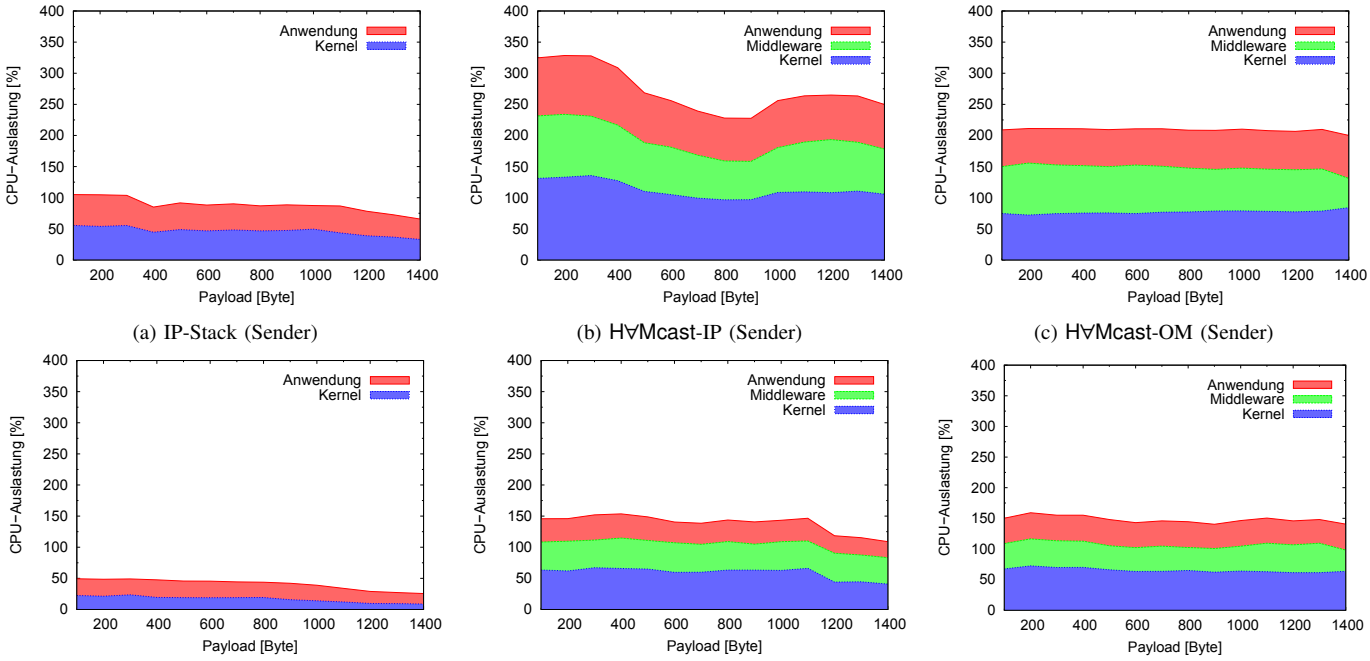
B. Diskussion

Die Ergebnisse der Performanz-Evaluierung des H∀Mcast-Prototypen haben gezeigt, dass die Middleware unter Verwendung des IP-Moduls in der Lage ist, für Pakete mit Payload von mindestens 500 Bytes eine 1 GBit/s Netzwerkverbindung vollständig auszulasten. Auch mit H∀Mcast-OM kann die Middleware bei insgesamt konstantem Paketdurchsatz ab einer Payload von 1000 B ca. 75 % des Datendurchsatzes von IP-Multicast erreichen. Die Performanz steht dabei einerseits im Zusammenhang zur Hardware, so dass auch der Linux IP-Stack (Abb. 3) für Paketgrößen bis 500 Bytes eine 1 GBit/s Netzwerkverbindung nicht vollständig auslasten kann. Andererseits wird sie auch vom maximalen Paketdurchsatz der Middleware in Kombination mit der IPC begrenzt (Abb. 3a). Die Paketgröße hat hier keinen wesentlichen Einfluss. Die CPU-Auslastung korreliert direkt mit den verarbeiteten Paketzahlen. Entsprechend sinkt sie für IP mit wachsender Paketgröße und bleibt für H∀Mcast-OM bei konstantem Durchsatz stabil. Das deutliche Absinken der CPU-Auslastung ab 400 B hängt mit der Netzwerkbandbreite zusammen, welche den maximalen Durchsatz begrenzt. Der Anstieg der CPU-Auslastung bei H∀Mcast-IP auf Senderseite (Abb. 4b) im Bereich 1000 – 1400 B wird durch die IPC verursacht.

Betrachtet man realistische Szenarien zur Gruppenkommunikation ist zu bemerken, dass durchsatzlastige Anwendungen wie Videostreaming meist Pakete mit hoher Payload, d.h. $>$ 1000 Byte, verwenden. Andere Anwendungen wie Nachrichtendienste und Online-Spiele versenden weniger Daten und haben einen geringeren Durchsatz. Die Ergebnisse der Evaluierung zeigen, dass der H∀Mcast-Prototyp für diese Anwendungsfälle geeignet ist.

VI. Zusammenfassung und Ausblick

In diesem Beitrag haben wir H∀Mcast, die Architektur eines universellen Multicast-Dienstes vorgestellt. H∀Mcast kombiniert eine namensorientierte Multicast-API mit einer Abstraktionsschicht in Form einer Middleware-Komponente. Hierdurch wird Gruppenkommunikation technologietransparent und domänenübergreifend möglich. Neben dem zugrundeliegenden Architekturkonzept haben wir die prototypische Implementierung von H∀Mcast diskutiert und die Systemperformanz im Detail untersucht. Für die Evaluierung wurde der



(a) IP-Stack (Sender) (b) H∀Mcast-IP (Sender) (c) H∀Mcast-OM (Sender)

(d) IP-Stack (Empfänger) (e) H∀Mcast-IP (Empfänger) (f) H∀Mcast-OM (Empfänger)

Abbildung 4. Vergleich der CPU-Auslastung beim Sender (oben) und Empfänger (unten) aufgeteilt in die jeweiligen Anteile für Anwendung, Middleware (bei H∀Mcast) und Kernel (100 % = 1 CPU).

H∀Mcast-Prototyp auf aktuellen Endsystemen mit dem IP-Stack des Linux-Kernels verglichen. Anhand der Ergebnisse konnte gezeigt werden, dass trotz der Zusatzlast für die Abstraktionsschicht die Performanz der Middleware auf aktuellen Endsystemen mit dem IP-Stack vergleichbar ist und somit auch für datenlastige Gruppenanwendungen wie Videostreaming geeignet ist.

Gegenwärtige Arbeiten untersuchen die Leistung und das Verhalten der H∀Mcast-Architektur in komplexen, hybriden Anwendungsszenarien und Netzwerktopologien in den verteilten Testbeds von German-Lab und Planet-Lab. Künftig sollen zusätzliche Multicast-Technologien, sowohl in virtualisierten Infrastrukturen als auch aus neuen Publish/Subscribe-Ansätzen in Form von Modulen für die H∀Mcast-Middleware implementiert und erprobt werden. Es ist das Ziel, so eine breite, vergleichsfähige Untersuchungsbasis für künftige Netzwerkkonzepte zu erhalten.

Danksagung

Wir danken Fabian Holler, Sebastian Wölke und Sebastian Zagaria für die Unterstützung bei der Umsetzung des Prototypen. Diese Arbeit wird vom Bundesministerium für Bildung und Forschung im Rahmen des Projekts H∀Mcast und der G-Lab Initiative gefördert, s. http://hamcast.realmv6.org.

Literatur

[1] S. E. Deering and D. R. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Trans. Comput. Syst.*, vol. 8, no. 2, pp. 85–110, 1990.

[2] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture," *IEEE Network Magazine*, vol. 14, no. 1, pp. 78–88, 2000.

[3] D. Thaler, M. Talwar, A. Aggarwal, L. Vicisano, T. Pusateri, and T. Morin, "Automatic IP Multicast Tunneling," IETF, Internet-Draft – work in progress 11, July 2011.

[4] S. Ratnasamy, M. Handley, R. M. Karp, and S. Shenker, "Application-Level Multicast Using Content-Addressable Networks," in *Proc. of 3rd Intern. Workshop on Network Group Communication (NGC'01)*, ser. LNCS, vol. 2233. London, UK: Springer–Verlag, Nov. 2001, pp. 14–29.

[5] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *IEEE JSAC*, vol. 20, no. 8, pp. 100–110, 2002.

[6] M. Wählisch, T. C. Schmidt, and G. Wittenburg, "BIDIR-SAM: Large-Scale Content Distribution in Structured Overlay Networks," in *Proc. of the 34th IEEE LCN*, NJ, USA: IEEE Press, Oct. 2009, pp. 372–375.

[7] X. Jin, K.-L. Cheng, and S.-H. G. Chan, "Island multicast: combining IP multicast with overlay data distribution," *IEEE Transactions on Multimedia*, vol. 11, no. 5, pp. 1024–1036, 2009.

[8] S. Lu, J. Wang, G. Yang, and C. Guo, "SHM: Scalable and Backbone Topology-Aware Hybrid Multicast," in *16th Intern. Conf. on Computer Communications and Networks (ICCCN'07)*, August 2007, pp. 699–703.

[9] B. Zhang, W. Wang, S. Jamin, D. Massey, and L. Zhang, "Universal IP multicast delivery," *Computer Networks*, vol. 50 (6), pp. 781–806, 2006.

[10] M. Wählisch and T. C. Schmidt, "Between Underlay and Overlay: On Deployable, Efficient, Mobility-agnostic Group Communication Services," *Internet Research*, vol. 17, no. 5, pp. 519–534, Nov. 2007.

[11] ——, "An a Priori Estimator for the Delay Distribution in Global Hyrid Multicast," in *Proc. of the ACM SIGCOMM CoNEXT. Student Workshop*. New York: ACM, Dec. 2009, pp. 19–20.

[12] D. Trossen, M. Sarela, and K. Sollins, "Arguments for an Information-Centric Internetworking Architecture," *SIGCOMM CCR*, vol. 40, pp. 26–33, April 2010.

[13] T. C. Schmidt and M. Wählisch, "Why We Shouldn't Forget Multicast in Name-oriented Publish/Subscribe," Open Archive: arXiv.org, Technical Report arXiv:1201.0349v1, 2012.

[14] S. Meiling, D. Charousset, T. C. Schmidt, and M. Wählisch, "System-assisted Service Evolution for a Future Internet – The HAMcast Approach to Pervasive Multicast," in *Proc. of IEEE GLOBECOM 2010, Workshop MCS 2010*. NJ, USA: IEEE Press, Dec. 2010, pp. 913–917.

[15] M. Waehlisch, T. Schmidt, and S. Venaas, "A Common API for Transparent Hybrid Multicast," IETF-Draft – work in progress 03, July 2011.