Source Routed Multicast
Ein Multicast-Routing-Header für IP Version 6

Mark Doll

Institut für Telematik, Universität Karlsruhe (TH)

Zusammenfassung. Gruppenkommunikation auf Basis von IP-Multicast nutzt die verfügbare Übertragungskapazität am effizientesten, benötigt jedoch in den Routern Zustand per-flow und skaliert daher nur begrenzt. Während für sehr große Gruppen der Effizienzvorteil den Aufwand der Zustandshaltung rechtfertigt, kann für kleinere Gruppen das IP-Multicastrouting dadurch ersetzt werden, dass man es mittels Application Layer bzw. Overlay Multicast in der Anwendungsschicht nachbildet. Oder man führt den zum Routing benötigten Zustand in jedem IP-Paket mit, so wie es Explicit Multicast und das in diesem Beitrag vorgestellte Multicast Source Routing tun. Der dafür neu spezifizierte Multicast Routing Header für IPv6 erlaubt einer Multicastquelle, ihre IP-Pakete über einen ähnlich optimalen Verteilbaum zu leiten wie es nativer IP-Multicast tun würde. Der Multicast Routing Header kodiert den Multicastroutingzustand platzsparend, schleifenfrei, erhält die Semantik des „Segments Left"-Felds und erlaubt in den Routern eine ähnlich einfache und effiziente Implementierung der Paketweiterleitung wie bei Unicast Source Routing mit im wesentlichen konstantem Aufwand je Segment bzw. dupliziertem IP-Paket.

1 Motivation

IP-Multicast ist zwar skalierbar im Sinne, dass es sehr große Gruppen unterstützt, ist jedoch bei der Skalierbarkeit im Sinne von vielen gleichzeitig aktiven Gruppen beschränkt. Denn während bei Unicast der Zustand in den Routern unabhängig von der Nutzung, also unabhängig von der Zahl an aktiven Datenströmen ist, benötigen Router bei IP-Multicast für die Paketweiterleitung Zustand pro Gruppe bzw. per-flow. IP-Multicast bleibt damit vorwiegend den wenigen großen bis sehr großen Gruppen mit hundert(tausend)en von Gruppenmitgliedern vorbehalten, bei denen der Effizienzvorteil den Aufwand der Zustandshaltung rechtfertigt. Aber auch kleinere Gruppen, wie sie z. B. bei Audio-/Videoconferencing, Computer-aided Collaborative Work, Teleteaching etc. auftreten, können entscheidend vom Effizienzgewinn durch Multicast profitieren, insbesondere dann, wenn eine schmalbandige Anbindung der Teilnehmer die Anzahl an gleichzeitig möglichen Unicastverbindungen und damit die maximale Anzahl an Teilnehmern bzw. die maximal erreichbare Übertragungsrate pro Teilnehmer und damit die Qualität limitiert.

Es gibt zwei grundsätzlich unterschiedliche Herangehensweisen, wie Multicast für kleinere Gruppen bereitgestellt werden kann: Entweder das Multicastrouting wird auf Anwendungsebene in die Endsysteme verlagert (Application Layer Multicast), wobei ggf. zusätzliche Proxies die Effizienz verbessern helfen (Overlay Multicast). Die vorgeschlagenen Verfahren sind vielfältig, zu den bekannteren gehören ALMI, NARADA, NICE, OMNI oder TOMA. Oder die Quelle benennt explizit alle Empfänger mit ihren Unicastadresssen im IP-Paket (Explicit Multicast). Nach diesem Prinzip arbeiten Verfahren wie Xcast [1], SIM [2] oder MSC [3], einen Überblick gibt [4].

Application Layer Multicast hat verglichen mit IP-Multicast einen höheren Bandbreitenverbrauch und normalerweise nicht dieselbe geringe Ende-zu-Ende-Verzögerung, denn IP-Pakete müssen je nach Lage der Endsysteme längere Pfade nehmen und fließen entlang einiger Pfade mehrfach bzw. parallel in mehrfacher Kopie. Darüber hinaus vergrößern die bei der Weiterleitung auf Anwendungsebene anfallenden Bearbeitungszeiten die Ende-zu-Ende-Verzögerung zusätzlich. Das Hauptproblem von Application Layer Multicast liegt jedoch in der Dynamik der Teilnehmer, da Endsysteme anders als Router ständig kommen und gehen können. Der durch Endsysteme aufgespannte Verteilbaum, über die Multicastpakete dupliziert und weitergeleitet werden, muss ständig angepasst werden, was immer wieder zu erhöhtem Paketverlust oder gar zeitweise völligem Konnektivitätsverlust führt. Overlay Multicast kann dieses Problem durch dedizierte Proxies vermeiden, die wie Router dauerhaft verfügbar sind. Damit entfällt jedoch der Vorteil, ohne explizite Unterstützung seitens des Netzes auszukommen.

Die Unterstützung durch die Router ist der Hauptnachteil von Explicit Multicast, bei dem die Router die im IP-Paket mitgeführte Liste von Empfängern auswerten können müssen, um das IP-Paket entsprechend zu duplizieren und weiterzuleiten. Der große Vorteil allerdings ist, dass sich damit dieselbe geringe Ende-zu-Ende-Verzögerung bzw. im Falle von Diffserv dieselbe Dienstgüte erreichen lässt wie mit nativem IP-Multicast, da die IP-Pakete ebenfalls die kürzesten Pfade nutzen können, und von den Routern direkt in der Ebene der Paketweiterleitung bearbeitet werden anstatt ineffizient auf Anwendungsebene. Der Bandbreitenverbrauch wird nur durch den Overhead der im IP-Paket mitgeführten Empfängerliste vergrößert, ist ansonsten aber vergleichbar gering wie bei nativem IP-Multicast.

Explicit-Multicast-Verfahren wie Xcast [1] transportieren lediglich eine Liste der Empfänger im IP-Paket mit. *Jeder* Router entlang des Pfades, der das Verfahren unterstützt, muss daher für *jede* der enthaltenen Adressen prüfen, über welchen Pfad bzw. Ausgangsinterface sie erreicht werden und ggf. das IP-Paket entsprechend duplizieren, wobei die Empfängerliste jeweils so modifiziert wird, dass nur noch die über das Ausgangsinterface erreichte Empfänger in der Liste verbleiben. Der Nachteil der Zustandshaltung von IP-Multicast wird bei Explicit Multicast also gegen einen deutlich erhöhten Bearbeitungsaufwand eingetauscht. SIM [2] versucht diesen zu reduzieren, indem es nur die verbliebenen Empfänger in einer Bitmaske markiert, was aufwendige Paketmodifikationen beim Weiterleiten erspart. Zudem verringern Caches in den Verzweigungsroutern den Bearbeitungsaufwand, erhöhen damit aber wiederum die Zustandshaltung in den Routern (wenn auch nur in den Verzweigungsroutern), die man gerade vermeiden wollte. Einen anderen Weg geht MSC [3], bei dem die im IP-Paket genannten Adressen nicht nur Empfänger sondern (auch) Router kennzeichnen, die dann per nativem IP-Multicast die lokalen Empfänger beliefern. Dazu wird zusätzlich die Multicastgruppe im IP-Paket vermerkt. Nur diese Router duplizieren IP-Pakete, wodurch sich der Bearbeitungsaufwand zwar nicht verringert, dafür aber die Zahl der Router, die einen erhöhten Bearbeitungsaufwand haben. Erkauft wird dies durch die ggf. nicht mehr optimalen Pfade und entsprechend größere Ende-zu-Ende-Verzögerung und größeren Bandbreitenverbrauch.

Der im folgenden vorgestellte Multicast Routing Header vermeidet den höheren Bearbeitungsaufwand von Explicit Multicast, indem die Quelle den Verteilbaum vorgibt (Source Routing). Erkauft wird dies mit im Worst Case doppelt so großem Overhead durch die zusätzlich mitzuführenden Adressen der Verzweigungsrouter, da im Worst Case je Empfänger ein innerer Verzweigungsrouter hinzukommt. Der Overhead kann halbiert werden, wenn nur die ersten 64 Bit jeder Routeradresse gespeichert werden, was ausreicht, wenn die Router statt über ihre Unicastadresse über ihre Subnet Router

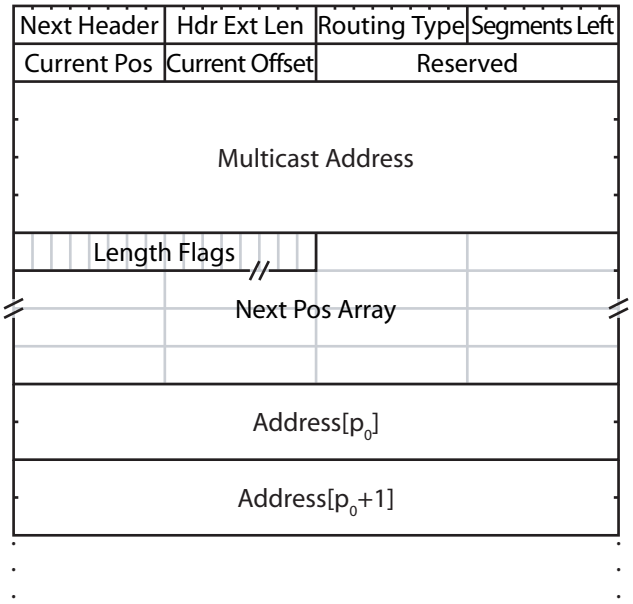

Abb. 1. IPv6 Multicast Routing Header.

Anycast-Adresse, die nach [5] jeder Router unterstützen muss, adressiert werden (wenn auch dann nicht mehr eindeutig). Voraussetzung ist, dass das Subnetzpräfix max. 64 bit lang ist. Wie eine stichprobenartige Überprüfung einer Reihe von Traceroutes ergeben hat, sind derzeit etwa ein Drittel aller Transitnetze im Internet /64er-Netze.

2 Source Routed Multicast

Beim (Loose) Source Routing gibt die Quelle die Zwischenstationen vor, über die das IP-Paket geleitet werden soll. Während bei Unicast hierfür eine Liste von Adressen genügt, die in der gegebenen Reihenfolge abgelaufen werden, muss für Multicast zusätzlich die Vater-Kind-Beziehung im Verteilbaum kodiert werden. Hierfür besitzt der Multicast Routing Header (Abb. 1) das *Next Pos*-Array.

Die Empfänger sind normalerweise Router, die dann die IP-Pakete per nativem IP-Multicast an die im Feld *Multicast Address* abgelegte eigentliche Multicastadresse und damit an die eigentlichen Empfänger weiterleiten. Aber auch wenn per Unicast direkt bis zu den Empfängern geroutet wird, ist *Multicast Address* nötig, um das IP-Paket dieser Multicastadresse zuordnen zu können. IP-Pakete werden von der Anwendung auf dem Quellhost mit der Multicastadresse als Zieladresse versendet und erreichen genauso die Anwendung auf Empfängerseite.

Current Pos gibt den Einstiegspunkt p in das *Next Pos*-Array an und *Current Offset* den zusätzlichen Offset o im *Address*-Array. Die Elemente von *Next Pos* sind jeweils ein Oktett lang, die von *Address* 8 Oktette. *Length Flag*, *Next Pos* und *Address* beginnen alle an derselben Stelle, so dass *Length Flag*[0], *Next Pos*[0] und *Address*[0] alle direkt hinter *Multicast Address* beginnen. Dadurch ist jeweils das erste Achtel von *Next Pos* und *Address* nicht nutzbar, da sich dort *Length Flag* bzw. *Next Pos* befinden. Der erste

gültige Offset ist damit $p_0 = \lceil N/7 \rceil$, wenn N die Anzahl aller Knoten im Verteilbaum ist. Die ersten p_0 Bits von *Length Flag* sind also ungenutzt und das letzte Element liegt entsprechend nicht bei $N - 1$ sondern bei $N - 1 + p_0$. Die letzten Bits bzw. Oktette von *Length Flag* und *Next Pos* sind ebenfalls ungenutzt, da Elemente von *Next Pos* immer an Oktettgrenzen und Elemente von *Address* immer bei Vielfachen von 8 Oktetten beginnen. Ist das p-te Bit im Feld *Length Flag*[p] gesetzt, sind für die p-te Adresse 128 bit im *Address*-Array abgelegt (sie nimmt dann zwei aufeinanderfolgende Felder des *Address*-Arrays ein), sonst 64 bit. Sie wird dann mit Nullen auf 128 bit zur Subnet Router Anycast-Adresse aufgefüllt.

Abb. 2 zeigt den Weiterleitungsalgorithmus ohne Fehlerbehandlung (ungültige/inkonsistente Felder etc.). Empfängt ein Router ein IP-Paket mit Multicast Routing Header, liest er beginnend mit *Next Pos*[p] der Reihe nach die Elemente von *Next Pos* ein, berechnet aus ihnen die neuen Werte für *Current Pos*, *Current Offset* und *Segments Left* und sendet für jedes eingelesene Element ein Duplikat des IP-Pakets an die an der Stelle *Address*[$p + o$] beginnende Adresse. Die Werte in *Next Pos* zeigen auf die Position, ab der der nächste Router, an den das Duplikat weitergeleitet wird, beginnen soll, *Next Pos* auszulesen. Da keine absolute Position sondern nur die Differenz von der des Elements zu der für den folgenden Router vorgesehenen Startposition abgelegt ist, kann nicht zu einem früheren Element zurückverwiesen werden, so dass Schleifenfreiheit garantiert ist. Ein Wert von Null bedeutet, dass es keine Folgeposition mehr gibt, der Router, an den das IP-Paket weitergeleitet wird, ist dann ein Blatt im Verteilbaum und wertet *Next Pos* nicht mehr aus. Um den Weiterleitungsalgorithmus einfach zu halten, erwartet jeder Router in *Next Pos* in seiner ab p beginnenden Kinderliste, dass zuerst die Blattknoten (Nullen) und dann die Zeiger für die Nichtblattknoten in aufsteigender Reihenfolge abgelegt sind.

Die ersten vier Felder haben dieselbe Bedeutung wie beim Routing Header Type 0 [6]. Insbesondere behält *Segments Left* seine Semantik. Es gibt zu jedem Zeitpunkt die Anzahl der verbliebenen Segmente im noch folgenden Teilbaum an (der, in dem der aktuelle Router die Wurzel darstellt), also die Anzahl der durch alle nachfolgenden Router zusammen noch erzeugten Duplikate.

Als letztes tragen alle Router *Multicast Address* als Zieladresse ein und leiten das IP-Paket per nativem IP-Multicast weiter. Verzweigungsrouter im Inneren des Verteilbaums werden üblicherweise keine Gruppenmitglieder haben, das IP-Paket wird dann von ihrem Multicastrouting einfach verworfen.

3 Anwendbarkeit

Die Quelle muss zwecks Wahl der Verzweigungsrouter die Pfade von sich zu jedem Empfänger bestimmen. Der Verzweigungsrouter für einen neuen Empfänger kann beispielsweise ähnlich Traceroute, aber mittels Intervallhalbierungsverfahren statt schrittweisem Inkrementieren des Hop Limits, mit wenigen Testpaketen gefunden werden. Unterstützt ein als Verzweigungsrouter vorgesehener Router den Multicast Routing Header nicht, sendet dieser ein ICMP Parameter Problem Code 0 an die Quelle zurück [6], so dass diese dann einen anderen Router als Verzweigungsrouter auswählen kann. Wird der Multicast Routing Header nur von wenigen Routern unterstützt, ist es sinnvoller, im Voraus zu ermitteln, ob ein avisierter Router den Multicast Routing Header unterstützt, beispielsweise durch Einträge SRV Record im DNS.

Die direkte Adressierung der Verzweigungsrouter bringt einige Nachteile mit sich. So sind die Pfade nach einer Routingänderung evtl. nicht mehr optimal. Je nach Dauer

einer Multicastsession sollte daher die Quelle periodisch die Pfade überprüfen und den Verteilbaum ggf. optimieren. Der Ausfall eines Verzweigungsrouters führt zu Konnektivitätsverlust, sofern ihn durch die Subnet Router Anycast-Adresse nicht ein anderer Router am selben Subnetz ersetzen kann. Der Ausfall wird durch ICMP Destination unreachable jedoch schnell erkannt und die Quelle kann dann zur schnellen Wiederherstellung der Konnektivität die Kinder des ausgefallenen Routers einfach an dessen Vater im Baum hängen.

Theoretisch lassen sich Verteilbäume mit bis zu 221 64 bit oder 110 128 bit-Adressen kodieren. Nimmt man die minimal zu unterstützende MSS von 1280 Oktetten als IP-Paketgröße an und lässt max. 50 % Overhead zu, lassen sich immerhin noch 62 bzw. 31 Adressen im Multicast Routing Header kodieren, was für viele Zwecke ausreichen dürfte.

Der Hauptvorteil des Multicast Source Routing gegenüber Explicit Multicast liegt in seiner einfachen und effizient implementierbaren Paketweiterleitung. Allerdings ist der Aufwand pro erzeugtem Duplikat nicht völlig konstant, da das Zählen der gesetzten *Length Flag*s von der Größe des folgenden Teilbaums abhängig ist. Sollte dies ein Problem darstellen, könnten die Elemente von *Next Pos* in zwei 4 Bit lange Zahlen aufgeteilt werden. Eine zeigt wie bisher die Differenz zur nächsten Startposition, die andere die Anzahl von 128 Bit-Adressen im davon übersprungenen Bereich. Hierdurch reduziert sich allerdings die maximal mögliche Größe eines Teilbaums auf 15 Knoten, so dass ab einer Verteilbaumgröße von 32 Knoten nicht mehr alle möglichen Bäume darstellbar sind.

Gemäß dem Motto des Internets „Keep it simple, stupid!“, vermeidet der Multicast Routing Header jede Komplexität in den Routern und bürdet stattdessen diese der Quelle auf – die entscheidende Vorbedingung, ohne die eine weitverbreitete Unterstützung in den Routern niemals denkbar wäre und ohne die der Multicast Routing Header nur von geringem Nutzen bliebe.

Literaturverzeichnis

1. Rick Boivie, Nancy Feldman, Yuji Imai, Wim Livens, Dirk Ooms, and Olivier Paridaens. Explicit Multicast (Xcast) Basic Specification. draft-ooms-xcast-basic-spec-00.txt, December 2000.
2. Vasaka Visoottiviseth. Sender-Initiated Multicast for small group communications. PhD thesis, Graduate School of Information Science Nara Institute of Science and Technology, January 2003.
3. Stefan Egger and Torsten Braun. Multicast for Small Conferences – A Scalable Multicast Mechanism Based on IPv6. IEEE Communications Magazine, 42(1):121–126, January 2004.
4. Explicit Multicast. http://www.watersprings.org/links/xcast/.
5. R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291 (Draft Standard), February 2006.
6. S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998.

Inhaltliche Korrekturen gegenüber Konferenzbandversion

Abb. 2, Zeile 17: „$p_{max} \leftarrow p + m$“ (war: „$p_{max} = m$“).
Zeile 34: „$l \leftarrow l - \mathit{SegmentsLeft} - 1$“ (war: „$l \leftarrow l - 1$“).
Zeile 36: Kommentar „{assertion: $p = p_{max}$ and $l = 0$}“ hinzugefügt.

```
if Segments Left = 0 and Current Pos = 0 then
  proceed with next header {skip this header}
end if
l ← Segments Left
p ← Current Pos
o ← Current Offset
m ← Next Pos[p]
while l > 0 and m = 0 do
  Segments Left ← 0
  set_destination_and_send
  p ← p + 1
  l ← l − 1
  m ← Next Pos[p]
end while
p2 ← p
o2 ← o
p_max ← p + m
while l > 0 and p < p_max do
  Current Pos ← p + m
  while p2 < Current Pos do
    o2 ← o2 + Length Flag[p2]
    p2 ← p2 + 1
  end while
  Current Offset ← o + o2
  if p + 1 < p_max then
    n ← m
    m ← Next Pos[p + 1]
    Segments Left ← m − n + 1 {child gets all till start of next child's subtree}
  else
    Segments Left ← l − 1 {last child gets all what's left}
  end if
  set_destination_and_send
  p ← p + 1
  l ← l − Segments Left − 1
end while
{assertion: p = p_max and l = 0}
Segments Left ← 0
Current Pos ← 0
Destination Address ← Multicast Address
resubmit packet to the IPv6 forwarding

macro set_destination_and_send
  first 8 octets of Destination Address ← Address[p + o]
  if Length Flag[p] = 0 then
    last 8 octets of Destination Address ← 0
  else
    o ← o + 1
    last 8 octets of Destination Address ← Address[p + o]
  end if
  resubmit a copy of the packet to the IPv6 forwarding
end macro
```

Abb. 2. Weiterleitungsalgorithmus für den IPv6 Multicast Routing Header.