Active QoS Routing

Michael Welzl[1], Alfred Cihal[1] und Max Mühlhäuser[2]

[1] Johannes Kepler Universität Linz, Telekooperation
Altenberger Straße 69, A-4040 Linz, Österreich;
michael@tk.uni-linz.ac.at,
http://www.tk.uni-linz.ac.at/~michael
[2] Technische Universität Darmstadt, Telekooperation
Karolinenplatz 5, D-64289 Darmstadt, Deutschland

Zusammenfassung In Active Networks (AN) kann an jeden Netzknoten Code verschickt werden, der auf ausgewählte Pakete angewendet wird. So wird u.a. die Einführung neuer Protokollmechanismen wesentlich vereinfacht. Wir stellen für solche AN ein neues Verfahren namens Active-QoS-Routing (AQR) vor. Damit wird es ohne Änderung der verwendeten Routingprotokolle und ohne Erweiterung von deren Metriken möglich, QoS-basiertes Routing zu betreiben. Zusätzlich schlagen wir eine Erweiterung der Node-OS-Spezifikation innerhalb von AN vor, die es erlaubt, aus in Routern bereits vorhandenen Werten QoS-Parameter zu berechnen. Mittels Simulation konnten wir eine signifikant bessere Performance von AQR gegnüber Shortest Path Routing nachweisen.

1 Einleitung

Active Networks (AN) zeichnen sich dadurch aus, dass Knoten dieser Netzwerke Pakete nicht nur weiterleiten, sondern den Inhalt untersuchen und geforderte Berechnungen durchführen [1]. Eine der Zielsetzungen dabei ist es, den für normale Netzwerke üblichen langwierigen Normungsprozess zu umgehen. An die Stelle einer Einigung über verwendete Protokolle soll eine Abstimmung der verwendeten Modelle und Methoden treten [2].

Charakteristisch für AN ist die Unterteilung in drei Schichten: Aktive Anwendungen (AA), das sogenannte Execution Environment (EE) und, auf unterster Ebene, das Knoten-Betriebssystem (Node-OS). Entsprechend der Spezifikation der Node-OS Schnittstelle [3] stellt dieses Flows, einen Thread-Pool, einen Memory-Pool und Channels zur Verfügnung. Gemäß den in der Litaratur eingeführten Definitionen von AN ist es jedoch den aktiven Anwendungen nicht möglich, auf untere Schichten der Protokollhierarchie zuzugreifen. Mit Spawning Networks und dem Genesis Kernel [4] wurde ein alternatives Konzept vorgestellt, das stattdessen auf einem sogenannten virtuellen Overlay-Netzwerk beruht. Da aber eine Implementierung zur Zeit nicht verfügbar ist und auch einen tiefen Eingriff in bestehende Strukturen bedeuten würde, scheint dieses Konzept für das geplante Einsatzgebiet ungeeignet zu sein.

Diese Arbeit soll zeigen, dass mit einer geringen Änderung der Architektur von Active Networks ein aktives QoS Routing ohne Eingriff in bestehende Routing-Protokolle und ohne Erweiterung von Routing-Metriken möglich wird.

2 Vorgeschlagene Änderungen zur Node-OS-Spezifikation

Wir schlagen wie erwähnt vor, in Active Networks lesenden Zugriff auf untere Protokollschichten zu erlauben. Hierfür bietet sich aufgrund der Aufgabenteilung in der Hierarchie nur das Node-OS an, wenn auf eine saubere Implementierung geachtet werden soll.

Die erwähnten Channels sind Teil der Schnittstelle zwischen Node-OS und Execution Environment. Sie stellen eine Abstraktion von Netzwerkverbindungen für Execution Enviroments dar. Laut derzeit gültiger Spezifikation stellt ein Node-OS lediglich Funktionen für den Auf- und Abbau solcher Channels und zum Senden und Empfangen von Paketen zur Verfügung [3]. Es ist kein Mechanismus vorgesehen, um Eigenschaften eines Channels abzufragen.

Wir schlagen vor, eine Erweiterung der Spezifikation vorzunehmen:

1. Jedem Channel werden bei Erzeugung die Eigenschaften der zu Grunde liegenden physischen Netzwerkverbindung zugewiesen.
2. Es werden Funktionen zum Auslesen derjenigen Parameter bereitgestellt, die entsprechend den Eigenschaften der Netzwerkverbindung relevant sind.

Wichtig ist anzumerken, dass es sich bei diesen Parametern nicht um sogenannte QoS-Parameter handelt (wie z.B.: aktuell verfügbare Bandbreite, zu erwartende Paket-Verlust-Rate), die in verbreiteten Rechnernetzen neu zu ermitteln wären. Stattdessen sind die relevanten Werte - im Falle IP-basierter Netzwerke - bereits in der Managment Information Base (MIB) des Routers vorhanden. Beispiele dafür sind MTU und die Zähler für ifIn- und ifOutOctets, aus denen sich Dienstgütespezifikationen errechnen lassen.

Es wurden sogar schon EEs entwickelt, die einen Zugriff auf diese Werte erlauben. Bekanntes Beispiel hierfür ist das System „Smartpackets“, bei dem die MIB direkt ausgelesen werden kann. Dieses EE ist allerdings speziell für Netzwerk-Management-Aufgaben entwickelt worden und erlaubt MIB-Zugriffe nur für entsprechend autorisierte Netzwerk-Administratoren [5].

Nachfolgend wird skizziert, wie die Implementierung der vorgeschlagenen Erweiterung den in diesem Artikel vorgestellten neuen Ansatz für QoS-Routing ermöglicht. Durch Auswertung ausgewählter MIB-Parameter entlang eines Netzwerkpfades kann dieses QoS-Routing ohne Veränderung bestehender Routing-Mechanismen verwirklicht werden. Es sind auch keine Eingriffe in die derzeit gebräuchlichen Routing-Metriken notwendig.

3 Vorteile von Active Routing gegenüber klassischen Ansätzen

Für QoS-basiertes Routing werden im Allgemeinen zwei Methoden beschrieben: Die Erweiterung bzw. Änderung der verwendeten Routing Metriken und die Verwendung von Source Routing [6].

Die Mehrzahl der QoS-basierten Routing-Methoden benutzen die erforderlichen Dienstgüteparameter in Routing-Protokollen als link- oder verbindungsspezifische Werte zusätzlich zu den Link-Kosten. Dies bedarf einer netzwerkübergreifenden Normung; im Zuge dieser Normung müssen die Dienstgüteparameter sowie deren Berechnung festgelegt werden. In der Praxis lassen sich jedoch Eigenschaften wie die durchschnittlich verfügbare Bandbreite schwer allgemein definieren; die Funktion zu deren Berechnung hängt typischerweise sowohl vom zugrundeliegenden Netzwerk als auch von den Erfordernissen der Anwendung ab.

Source Routing als Basis für Dienstgüteunterstützung, wird beispielsweise in [6] oder in [7] an Hand einer Erweiterung des in OSPF verwendeten Dijkstra-Algorithmus beschrieben. Da AQR auf Source Routing basiert, wird auf die spezielle Problematik dieser Methode in einem späteren Kapitel eingegangen.

Mit unserem Ansatz auf der Basis von Active Networks können Dienstgüteparameter flexibel (z.B. anwendungsabhängig) definiert und berechnet werden. Eine Festlegung auf allgemein spezifizierte Dienstgüteparameter und deren Berechnung innerhalb des Node-OS ist nicht erforderlich. Das Node-OS stellt lediglich in Routern bereits vorhandene Werte zur weiteren Verarbeitung durch eine AA zur Verfügung. Dies schafft eine weitgehende Unabhängigkeit von der Art des zugrundeliegenden Netzwerkes. Es ist Aufgabe der AA, die gelieferten Werte entsprechend ihres Verwendungszweckes zu verarbeiten und zu interpretieren.

Wichtig ist also festzuhalten, dass Active-QoS-Routing nicht primär darauf abzielt, gegenüber bekannten Verfahren qualitative Verbesserungen im Mittel über alle Anwendungen zu erzielen. Vielmehr steht die Abkehr von in Routern fest implementierten Mechanismen zur Berechnung und Bereitstellung von QoS-Parametern im Vordergrund; aus Sicht spezifischer Anwendungsklassen und deren Interpretation bestimmter QoS-Parameter ist allerdings sehr wohl auch eine qualitative Verbesserung zu erwarten. Entsprechend dem Stand unserer Forschung beschränken wir uns nachfolgend allerdings darauf, den generellen Machbarkeitsnachweis zu liefern und die signifikante Verbesserung gegenüber klassischen IP-Routing aufzuzeigen.

4 Mechanismus

Active-QoS-Routing basiert auf der Annahme, dass im AN eine Link-State-Tabelle vorliegt wie von OSPF oder einem vergleichbaren Routing-Algorithmus generiert. Weiters müssen die zur Berechnung der Dienstgüte erforderlichen Informationen zur Verfügung stehen; dafür ist wie erwähnt lesender Zugriff auf untere Schichten normalerweise notwendig und hinreichend. Für den speziellen

Fall der Übertragungsverzögerung (Delay) kann dieser Lesezugriff sogar entfallen, da diese (zumindest bei über Timeserver synchronisierten Knoten) direkt aus der Systemzeit berechnet werden kann.

Eine Anwendung, die eine bestimmte QoS-Anforderung stellt, löst folgenden Prozess aus:

- Aus der Routing-Tabelle werden alle nicht-zyklischen Pfade zum Ziel berechnet.
- Entlang aller festgestellten Pfade werden Messpakete unter Verwendung von Multicast auf AN-Ebene verschickt.
- Jeder Knoten prüft eine im Messpaket enthaltene QoS-Mindestanforderung. Kann diese nicht erfüllt werden, wird das Paket verworfen.
- Der QoS-Parameter wird mit dem für den Knoten gültigen Wert verglichen und bei Bedarf korrigiert.
- Der Empfängerknoten erhält die Pakete, die die QoS-Mindestanforderung erfüllen und erstellt daraus bis zum Ablauf einer den Anforderungen entsprechend festzulegenden Zeitspanne eine Liste aller gültigen Pfade.
- Der Empfängerknoten wählt aus dieser Liste den Pfad mit dem besten QoS-Parameter aus und sendet diesen in einem Antwortpaket an den Ausgangsknoten zurück.

Dieser Prozess wird in bestimmten zeitlichen Abständen wiederholt, um auf geänderte Netzwerkzustände reagieren zu können.

5 Kritische Betrachtung und Fragestellungen

Offensichtlich ist bei der hier beschriebenen Methode im Besonderen, wie bei Verwendung von Source Routing im Allgemeinen, die Frage der Skalierbarkeit kritisch zu betrachten. Der Einsatz von AQR durch eine große Anzahl von Anwendungen kann die Begrenzung der Netzwerk- und Router-Belastung durch untere Schranken für die Aktualisierungsabstände erfordern. Eine andere Möglichkeit stellt die Beschränkung der Benutzung von AQR auf bestimmte Flows, z.B. solche mit langer Ausführungszeit, wie in [8] in anderem Zusammenhang beschrieben, dar. Im Vergleich mit QoS-Routing-Verfahren aus der Literatur ist allerdings anzumerken, dass auch diese — insbesondere aufgrund der erweiterten Metriken — zu einer Router-Mehrbelastung führen, deren Skalierbarkeit typischerweise noch nicht hinreichend gesichert ist; dasselbe gilt für die Netzwerkbelastung, da üblicherweise Daten über veränderte QoS-Parameter ausgetauscht werden.

Andererseits hat sich mit Link-State-Routing schon in klassischen IP-Netzen ein schlecht skalierbarer Routing-Ansatz (im Vergleich zu Distance-Vector-Routing) durchgesetzt, was bereits zu erheblichen Beschränkungen der Größe von Routing-Domänen führt. Es ist daher zu erwarten, dass ein pragmatischer Ansatz für die Randbedingungen des Einsatzes von AQR im Internet gefunden werden kann. Unsere bisherigen Simulationen bestärken diese Annahme. Numerische Werte zu Parameterschranken und Netz- bzw. Router-Belastungen können

in diesem Artikel allerdings noch nicht angegeben werden, da die hierfür erforderlichen umfangreiche Untersuchungen noch nicht abgeschlossen sind. In diese fließen u.a. folgende Fragestellungen zur Parametrisierung von AQR ein:

- In welchen Zeitabständen sollen die Messungen durchgeführt werden?
- Wann soll vom empfangenden Knoten das Antwortpaket gesendet werden?
- Sollen Messpakete über verbindungsorientierte oder -lose Protokolle versendet werden?

Die bisherigen Ergebnisse unserer Simulations-Testreihen zeigen, dass diese Fragen nicht leicht einheitlich beantwortet werden können. Vielmehr hängen auch diese Werte und Methoden von den Dienstgüteparametern und deren anwendungsabhängiger Spezifikation ab. Gerade diese Beobachtung unterstreicht andererseits die Sinnhaftigkeit unseres Ansatzes: sie zeigt die Notwendigkeit einer flexiblen QoS-Berechnung und rechtfertigt den Einsatz einer Lösung, die auf AN basiert.

6 Simulation von Active QoS Routing

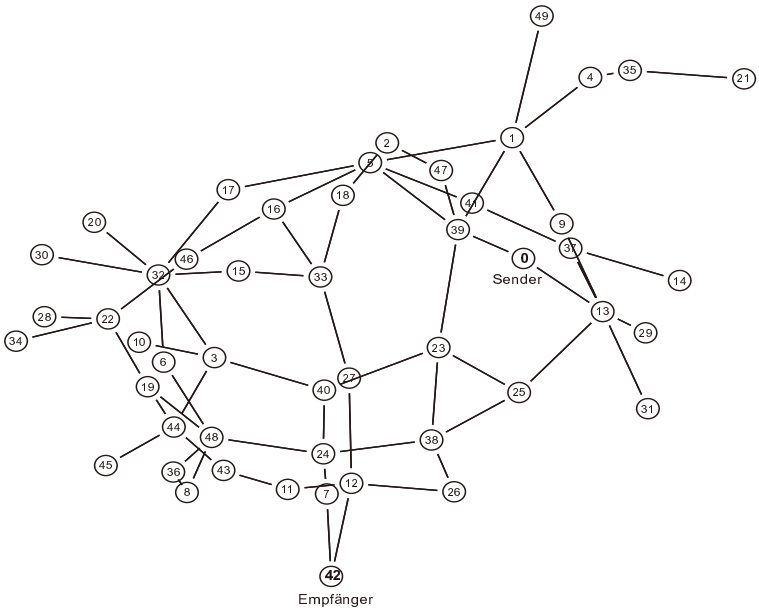

Abbildung 1. Verwendete Topologie

6.1 Verwendete Werkzeuge

Unsere Simulationen basieren auf dem Network-Simulator "ns" in Version 2.1b6. Die erforderlichen Änderungen und Erweiterungen werden als Quellcode auf [9] dokumentiert.

6.2 Netzwerktopologie

Für die hier diskutierte Testreihe wurde die in Abbildung 1 dargestellte Topologie mit 50 Knoten verwendet.

6.3 Der Algorithmus

Wir verzichten vorerst auf die Verwendung eine optimierten Algorithmus, um zu zeigen, dass auch mit einer sehr simplen Lösung ein entscheidender Vorteil gegenüber Shortest-Path-Routing erzielt werden kann.

Dabei wird kein Multicast verwendet, auf die Aufteilung des constant-bit-flows auf mehrere Pfade wird verzichtet. Für die Pfadsuche wird eine Erweiterung der Tiefensuche nach Sedgewick [10] eingesetzt.

Zum besseren Verständnis folgt eine Darstellung des Algorithmus in Pseudo-Code.

```
/*---------------------*/
/*   Initialisierung   */
/*---------------------*/

  for(i=0; i<maxV; i++)   // initialisieren der Adjazenzmatrix
    for(j=0; j<maxV; j++)
      if(link_exists_in_OSPF_table(i, j))
        a[i][j]=1;
      else
        a[i][j]=0;

  waveid = 0;                  // Nummer der Paket-Welle
  *bestPath = [Quelle, Ziel];  // Knoten haben eindeutige Nummern
  *bestPathCopy = [Quelle, Ziel];
  smallestDelay = GROESSTER MOEGLICHER WERT;
  max_delay = WERT;            // WERT wird von Anwendung uebergeben;

/*-----------------------------------*/
/*   Versenden einer Messpaket-Welle   */
/*-----------------------------------*/

void visit(int k, char path[], int val[]) {

  VERLAENGERE path UM k;

  if(k == destination) {
    send_packet(path);
    return;
  }

  int t;
  val[k] = 1;
  for(t=0; t<maxV; t++)
    if(a[k][t] != 0)
      if(val[t] == 0)
```

```
        visit(t, path, val);
}

send_packet(path) {

   new packet(pkt);
   pkt->time = NOW;              // Absendezeitpunkt
   pkt->path = path;             // der Strict-Source-Routing-Pfad
   pkt->pathcopy = path;         // Kopie des vollstaendigen Pfads fuer Empfaenger
   pkt->waveid = waveid;         // fortlaufende Nummer der Paket-"Welle"
   pkt->code = {                 // Der Active-Code
      if(NOW - pkt->time < max_delay) {
         ENTFERNE ERSTEN EINTRAG IN path;
         send(pkt, dest=erster path-eintrag);
      }
      else
         drop(pkt);

   send(pkt, dest=ERSTER EINTRAG IN path);
}

KOPIERE bestPath NACH bestPathCopy;
smallestDelay = GROESSTER MOEGLICHER WERT;    // Reinitialisierung

int val[maxV];
for(i=0; i<maxV; i++) val[i]=0;
char path[maxPath] = "";
visit(atoi(argv[2]), path, val);
waveid++;

/*-----------------------------------*/
/*   Herstellung des Antwortpaketes   */
/*-----------------------------------*/

recv(pkt) {

        new packet(replypkt);
        replypkt->delay = NOW-pkt->time;       // berechnetes Gesamtdelay
        replypkt->path = pkt->pathcopy;        // vollstaendiger Pfad
        replypkt->waveid = pkt->waveid;        // Paket-Welle
        send(replypkt, dest = source of pkt);
}

/*-----------------------------------*/
/*   Sender verarbeitet Antwortpaket   */
/*-----------------------------------*/

recv(replypkt) {

        if( (replypkt->delay < smallestDelay) && (replypkt->waveid == waveid) ) {
           smallestDelay = replypkt->delay;
           KOPIERE replypkt->path NACH bestpath;
        }
}
```

6.4 Ergebnis

Trotz des Verzichts auf optimierende Maßnahmen konnte ein gegenüber Shortest-Path-Routing um durchschnittlich 20 % kürzeres Delay erreicht werden. Für die in Abbildung 2 dargestellten Simulationsergebnisse wurden folgende Einstellungen verwendet:

- Zeit zwischen den Messpaketwellen: 3,5 Sekunden
- Störverkehr: durchschnittlicher TCP-Verkehr

– nominelle Bandbreite: 1 Mb im gesammten Netzwerk
– Senderknoten: 0
– Empfängerknoten 42

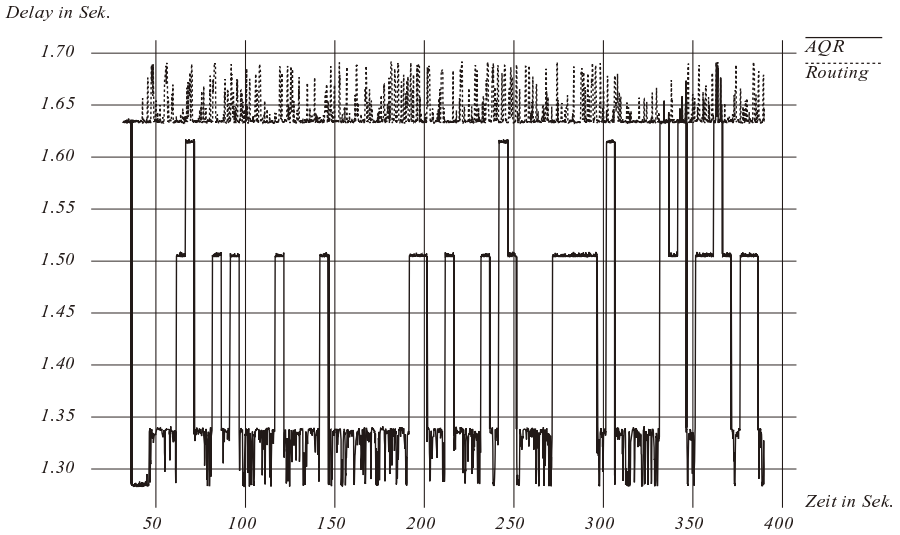

Abbildung 2. AQR versus Shortest-Path-Routing

6.5 Der erweiterte Algorithmus

Wir haben folgende Änderungen am Algorithmus durchgeführt:

– Verwendung von Multicast für Messpakete zur Reduktion der Netzwerkbelastung
– Möglichkeit der Aufteilung des Nutzverkehrs auf mehrere Pfade

Von der Aufteilung des Nutzverkehrs erwarten wir geringere Schwankungen in der erzielten Dienstgüte, eine bessere Auslastung des Netzwerks und eine Verhinderung einer durch den Flow selbst ausgelösten Oszillation zwischen mehreren Pfaden. Die Aufteilung erfolgt durch Festlegen eines Toleranzwertes, um den der jeweilige QoS-Parameter über- bzw. unterschritten werden darf. Als erstes Ergebnis zeigt Abbildung 3 die Abhängigkeit des gemessenen durchschnittlichen Delays von der Wahl dieses Wertes. Es zeigte sich dabei ein deutlicher Anstieg des durchschnittlichen Delays bei Überschreitung von 8% Toleranz.

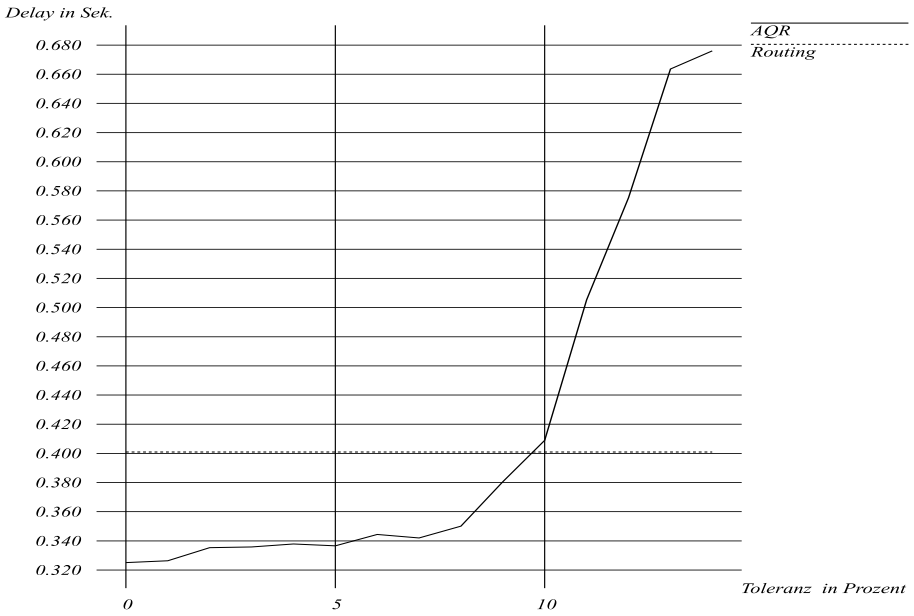

Abbildung 3. Delay bei verschiedenen Toleranzwerten

7 Schlussbemerkungen

Wir haben eine Möglichkeit vorgestellt, mittels Active Networks eine Form des QoS-basierten Routing zu verwirklichen. Die verwendete Methode hat wesentliche Vorteile gegenüber herkömmlichen Ansätzen: sie kommt ohne Änderung bestehender Routing-Protokolle und ohne zusätzliche Routing-Metriken aus; Als Voraussetzung für den breiten und allgemeinen Einsatz schlagen wir eine Änderung der Node-OS-Spezifikation vor, da für die Berechnung von QoS-Parametern typischerweise ein Zugriff auf verschiedene Kenngrößen des zugrundeliegenden Netzwerks notwendig ist.

Die diskutierten Simulationen bezüglich des QoS-Parameters „Delay“, welche wir durchführten, zeigten eine signifikant bessere Performance von AQR gegenüber Shortest-Path-Routing. Weitere Simulationen unter Verwendung des erweiterten Algorithmus und von weiteren QoS-Parametern sind durchzuführen. Diese sollen auch Aufschluß über Ausmaß und Randbedingungen der Skalierbarkeit bringen, ein für QoS-Routing allgemein noch nicht hinreichend untersuchtes Problem. Einige Antworten hierzu sind bis zur KIVS bereits zu erwarten.

Literatur

1. David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall and Gary J. Minden: *A Survey of Active Network Research* . IEEE Communications Magazine, Vol. 35, No. 1, pp80-86. January 1997

2. John Guttag: *Report on Active Network Activity.* Active Nets Workshop, March 1997, Baltimore, MD.
3. AN Node OS Working Group: *Node OS Interface Specification*, Edited by Larry Peterson, 24 Jänner 2000
4. A. Campbell, M. Kounavis, D. Villela, H. De Meer, K. Miki, J. Vicente: *The Genesis Kernel: A Virtual Network Operating System for Spawning Network Architectures*, Second IEEE Conference on Open Architectures and Network Programming, OPENARCH '99, New York, March 26-27, 1999.
5. B. Schwartz, A.W. Jackson, T. Strayer, W. Zhou, R.D. Rockwell and C.Partridge: *Smart Packets for Active Networks*, Second IEEE Conference on Open Architectures and Network Programming, OPENARCH '99, New York, March 26-27, 1999.
6. Z. Wang and J. Crowcroft. *Quality-of-Service Routing for Supporting Multimedia Applications.* IEEE JSAC, 14(7):1288–1234, September 1996.
7. R. Guerin and A. Orda and D. Williams. *QoS Routing Mechanisms and OSPF Extensions* IETF Internet Draft , November 1996.
8. Anees Shaikh, Jennifer Rexford, and Kang G. Shin. *Load-sensitive routing of long-lived ip flows.* In Proceedings of SIGCOMM, September 1999.
9. `http://www.tk.uni-linz.ac.at/~micky/AQR.html`
10. Robert Sedgewick: *Algorithmen*, Addison-Wessley (Deutschland) GmbH, 2. korrigierter Nachdruck 1995.