Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Lehrstuhl für Rechnernetze und Telematik

WS 2007/08

Seminararbeit

Routing Information Protocol (RIP), Open Shortest Path First (OSPF)

Martin Bauer

22. Januar 2008

Betreut durch Prof. Dr. rer. nat. Christian Schindelhauer

Abstract

Dem Routing Information Protocol (RIP) und dem Open Shortest Path First (OSPF) Protokoll liegen die beiden Konzepte der Distanz-Vektor-Algorithmen und der Link-State-Algorithmen zu Grunde. Beide suchen nach den kürzesten Pfaden in Netzwerken. Dabei nutzt ersteres eine verteilte Organisation der Netzwerktopologie. Zweiteres dagegen verteilt den vollständigen Datensatz auf alle Router und sorgt für deren Konsistenz.

Inhaltsverzeichnis

1 Einleitung **2**

2 Router Information Protokol **3**
2.1 Distanz Vektor Algorithmus 3
2.2 Implementierung RFC1058 (RIP) 4
2.3 Count-to-Infinity-Problem 5
2.4 Triggered Update 5
2.5 Split-Horizon mit Poisoned Reverse 6

3 Open Shortest Path First **6**
3.1 Link-State Algorithmus 6
3.2 Implementierung in RFC2328 (OSPF) 8
3.3 Multicast 8
3.4 Topologie 8
3.5 Sicherheit 9

4 Zusammenfassung **9**

1 Einleitung

Rechner nehmen immer mehr Einfluss in unseren Alltag, ob nun unmittelbar in Form von Personal Computer und Notebooks, oder weniger offensichtlich wie zum Beispiel in Autos und Haushaltsgeräten. Um das volle Potential zu erschließen, mt in Zukunft der Vernetzung - ob nun über Funk, Netzwerkkabel oder Stromnetz - eine zentrale Rolle zu.

Der Datentransport über Wegweiser (Router) via Protokollen wie dem Routing Internet Protocol (RIP) und dem Open Shortest Path First (OSPF) sichern dabei einen schnellen, effizienten und ausfallsicheren Ablauf zu.

Rechnernetzwerken liegt das Modell eines **Graphen** $G(V, E)$ zugrunde. Die Knotenmenge V stellt dabei die Menge der Router dar. Eine Kante $(x, y) \in E$ symbolisiert die direkte Verbindung von zwei Routern $x, y \in V$ Zwei Knoten die sich eine Kante teilen, nennt man Nachbarn. Es gilt: Kanten sind ungerichtet

$$(x, v) = (v, x)$$

und Knoten haben keine Kante zu sich selbst ($x \neq x$). Weiterhin besitzt jede Kante eine nicht-negative Kostenfunktion $c(x, v)$, welche in der Praxis durch Antwortzeit, Leitungslänge oder Bandweite bestimmt wird.

Im Folgenden soll das Problem betrachtet werden, einen Pfad mit minimalen Kosten von einem Startknoten x zu einem Zielknoten y zu finden.

2 Router Information Protokol

2.1 Distanz Vektor Algorithmus

Bellman[4] und Ford[5] schlagen einen verteilten Algorithmus vor, in dem jeder Knoten nur einen Ausschnitt des Graphen kennt und dadurch erst alle Knoten gemeinsam das vollständige Wissen über den Graphen erlangen. Zum einen kennt ein Knoten x die Kosten zu allen seinen Nachbarn $c(x, v), \quad v \in V$. Zum anderen erlangt er über sein Nachbar v die Kenntnis über weitere Knoten $d_v(y)$

Das Minimum aus der Menge der Wegkosten zum Nachbar und Wegkosten vom Nachbar zum Ziel ist als Pfad mit minimalen Kosten (kürzester Pfad) definiert.

$$d_x(y) = min_v\{c(x, v) + d_v(y)\}$$

Es wird deutlich, dass ein iterativer Aufruf von $d_v(y)$ für jeden Nachbarknoten v eine neue Potenzmenge über V generiert.

Am Beispiel von (Abb. 1) kann man sehen, dass der kürzeste Pfad von u nach z über x und y führt, da $d_y(z) = 2$ die kürzeste Verbindung von y zum Ziel ist. Im nächsten Schritt

stellt man fest, dass es für $d_x(z) = 3$ und schlußendlich auch für $d_u(z) = c(u,x) + d_x(z) = 4$ keinen kürzeren Pfad gibt.

Abb. 1

2.2 Implementierung RFC1058 (RIP)

In RFC1058[1] spezifizieren die Autoren ein Protokoll auf Basis des Bellman-Ford-Algorithmus. Die Kostenfunktion für jede Kante ist dabei konstant $c(x,v) = 1$. Die Knoten kommunizieren über das UDP Protokoll/Port 520 mit ihren Nachbarn. D.h. jede 30 Sekunden übermittelt ein Nachbarknoten v einen Vektor D_v bestehend aus allen ihm bekannten Zielknoten (y) und den dazugehörigen kürzesten Pfaden.

$$D_v = [D_v(y) : y \in V]$$

Der empfangende Knoten x kann daraus wie folgt seinen Vektor $D_x(y)$ generieren:

$$D_x(y) = min_v\{\underbrace{c(x,v)}_{1} + D_v(y)\}, \quad \forall y \in V$$

Der Knoten x vergleicht alle y-Komponenten des empfangenen Vektor mit seinem eigenen Vektor.

$$D_x(y) > 1 + D_v(y), \quad \forall y \in V$$

.

Zeigt nun ein Vergleich, dass neben der bisher bekannten Route $D_x(y)$ eine weitere Route über v existiert, die einschließlich der Kosten zu v kürzer ist, so wird diese ab sofort vermerkt.

Man kann nun sehen dass wenn n gleich der Anzahl aller Knoten $n = |\{V\}|$ ist, so müssen im worst-case $n-1$ Distanzen miteinander verglichen werden. Da dies auf alle n Knoten zutreffen kann, ist die Zeitkomplexität von der Größenordnung $O(n^2)$ in einem Graphen mit n Knoten.

Das Routing Information Protocol begrenzt die maximal möglichen Kosten auf 15. Der Wert 16 ist für den Status unerreichbar (infinity) reserviert. Dies hat weitreichende Folgen für die Struktur eines Graphen. So darf der maximale Abstand zweier Knoten nicht mehr als 15 Knoten übersteigen. Es bedeutet aber auch, dass durch die 30 sekündliche Übertragung eine Pfadänderung im worst-case bis zu 7,5 Minuten benötigt, bis der gesamte Graph konsistent ist. Man spricht in diesem Zusammenhang auch von Konvergenz.

2.3 Count-to-Infinity-Problem

Angenommen, es gibt drei Knoten x, y, z. Der Knoten y hat mit Kosten 1 je eine Kante zu x und z. Die Knoten x und z sind über y erreichbar und haben Kosten 2 zueinander und der Graph befindet sich in einem konvergenten Zustand (Abb. 2).

Abb. 2

Fällt nun die Kante (x, y) aus, so bemerkt Knoten y dies. Sendet aber zwischenzeitlich z seinen Distanzvektor mit $D_z(x) = 2$ an y, so glaubt dieser mit $D_y(x) = min_v\{c(y,z) + D_z(x)\} = 1 + 2 = 3$ zu erreichen und hat dadurch den eigenen Kantenausfall ohne es zu bemerken, revidiert. Als nächstes sendet nun y an z seinen Vektor $D_y(x) = 3$ sodass z wiederum seinen Eintrag um eins auf 4 erhöht. Der Austausch findet solange statt, bis einer der beiden die Kosten 16 erreicht und die Route als Unerreichbar (Infinity) vermerkt wird.

2.4 Triggered Update

In RFC1058 ist vereinbart, dass sich die Knoten im Abstand von 30 Sekunden ihre Distanzvektoren mitteilen. Im Fall des Count-to-Infinity können dadurch 7,5 Minuten

vergehen bis die Inkonsistenz behoben ist. Der in RFC2453[3] vorgeschlagene Triggered Updates Mechanismus sorgt nun dafür, dass Änderungen am Kantengewicht immer sofort übertragen werden, wodurch sich die Konvergenzzeit verringert.

2.5 Split-Horizon mit Poisoned Reverse

Auch wenn nun Änderungen im Graph zügiger vorgenommen werden, so wäre eine Verringerung des durch Count-to-Infinity entstehenden Verkehrs wünschenswert. Der Split-Horizon Mechanismus kann dieses Problem lösen, indem Knoten B beim nächsten Update an Knoten C die Unereichbarkeit sendet. Knoten C modifizert seinen Vektor, aber sendet das Ergebnis nicht mehr an Knoten B zurück.

Nun ist es aber immer noch möglich, dass beispielsweise ein Knoten Y, der kein direkter Nachbar von B ist, unerreichbar wird. Diesen kann Knoten Y nicht mehr explizit ausschließen. Deshalb werden alle über diese Schnittstelle erlernten Knoten für unerreichbar erklärt und wieder neu erlernt.

3 Open Shortest Path First

3.1 Link-State Algorithmus

Dem von Dijkstra [6] (Abb. 3) vorgeschlagenen Algorithmus liegt zugrunde, dass vorab der vollständige Graph $G(V, E)$ allen Knoten bekannt ist. Ein Knoten x soll die Kosten für den kürzesten Pfad $D(y)$ zu Knoten y ermitteln. Die dahinter stehende Idee ist es, einen Knoten v mit geringsten Kosten zu bestimmen und anschließend für alle Nachbarn y zu zu prüfen, ob y über v kürzer zu erreichen ist. Hat man solch einen Knoten v gefunden, hat man einen neuen kürzesten Pfad $D(y)$

$$D(y) = min(D(y), D(v) + c(v, y))$$

Für die Realisierung des Algorithmus wird eine Residualmenge N' benötigt, in welcher die Knoten gesammelt werden, für die der kürzeste Pfad bereits bekannt ist und weiterhin benötigt man für jeden Knoten v eine Funktion die seinen Vorgänger $p(v)$ im kürzesten Pfad vermerkt.

In der Initialisierungsphase beginnt der Algorithmus indem er alle kürzesten Pfade auf unendlich setzt.

$$D(v) = \infty, \quad \forall v \in Vx$$

Ausgenommen davon sind nur die Distanzen $D(v)$, welche den Nachbarn v zum Startknoten x haben. Diese erhalten als Startwert die Kantengewichte.

$$D(v) = c(x, v)$$

In der Aktualisierungsphase wird ein Knoten v gesucht der nicht in N' enthalten ist und den kürzesten Pfad $D(v) = min_v\{V\}$ besitzt. Für den gewählten Knoten werden die Pfadkosten $D(y)$ (wie in der Idee beschrieben) errechnet. Abschließend wird der Knoten v der Menge N' hinzugefügt. Phase 2 wird solange wiederholt bis auch der letzte Knoten verbraucht wurde.

Betrachtet man den Algorithmus unter den Gesichtspunkte der Zeitkomplexität, so stellt man fest, dass für die Wahl des kürzesten Pfad, in der ersten Iteration $n - 1$, in der zweiten $n - 2$, ...usw. mal gesucht werden muss. D.h. die Gesamtzahl der durchsuchten Knoten beträgt $n(n + 1)/2$, womit der Algorithmus in der Klasse $O(n^2)$ liegt. Als Alternative für die Suche in Mengen wurden Vorrangwarteschlangen [7] und Stapel vorgeschlagen, wodurch sich die Zeitkomplexität bis auf $O(n\ log\ n + |E|)$ drücken ließ.

Dijkstra-Algorithmus:

```
Initialization
N' = {x}
  for all nodes v
    if v is a neighbor of x
      then D(v) = c(x,v)
    else D(v) = infinity
Loop
  find v not in N' such that D(v) is a minimum
  add v to N'
  update D(v) for each neighbor v of w and not in N':
    D(y) = min(D(y), D(v) + c(v,y))
  /* new cost to y is ether old cost to y or known
  least path cost to v plus cost from v to y */
until N' = N
```

Abbildung 3

3.2 Implementierung in RFC2328 (OSPF)

Während RIP für den Einsatz in kleineren Netzen geeignet ist, haben die Autoren des RFC2328[2] das Protokoll OSPF für mittlere bis große Netze konzipiert. Auf Basis des Dijkstra-Algorithmus kennt jeder Knoten den vollständigen Graphen. Diesen erhält er in regelmäßigen Abständen in Form von so genannten Link-State-Advertisments (LS-Adv) von jedem Knoten im Graphen. In stabilen Netzen kann der zeitliche Abstand zwischen zwei LSAdv bis zu 30 Minuten betragen. OSPF verwendet dafür ein eigenständiges Protokoll (Protocol-ID 89) auf IP-Ebene und ist nicht wie RIP von UDP abhängig. Jeder Knoten überwacht die Kosten zu seinen Nachbarn mittels HELLO-Paketen. Die Kostenfunktion ist variabel und wird auch **Link State** genannt. Bemerkt ein Router Veränderungen am Link State, wird dies unverzüglich an alle anderen Knoten gesendet.

3.3 Multicast

Wenn Änderungen am Link State häufig eintreten, kann der Verkehr rasch zunehmen. Ein besonderes Problem ergibt sich in der Praxis dadurch, dass TCP/IP basierte Netze auf Punkt-zu-Punkt-Verbindungen basieren. In der Folge muss ein Knoten immer das gleich LSAdv-Paket an alle anderen Knoten senden. Abhilfe schafft hier Multicast. Multicast ermöglicht es ein Paket an mehrere Adressaten zu senden. Das Paket breitet sich dadurch flutartig aus und jede Kante wird nur einmal durchlaufen.

3.4 Topologie

Der OSPF-Graph hat die Eigenschaft, dass mit zunehmender Anzahl an Knoten der Verkehr Infolge der Flutungen überproportional zunimmt. OSPF bietet die Möglichkeit innerhalb des Graph einen Knoten zu bestimmen der über die beste Konnektivität verfügt und über den dann alle Knoten ihre LSAdv versenden. Als weiteren Schritt kann der Graph in Teilgraphen zerlegt werden, die dann über einen zentralen Backbone miteinander verbunden werden. Die bisher besprochenen Eigenschaften bleiben dadurch in den Teilgraphen erhalten. Die an den Grenzen befindlichen Knoten werden um die Aufgabe erweitert, auch Mitglied in benachbarten Graphen zu werden und alle dortigen Knoten zu lernen. Bei Bedarf können unbekannte Ziele über die Teilgraphen hinweg ermittelt werden. Kann der Grenzknoten kein Ziel in seiner Tabelle finden, leitet er dass Paket

an die Backbone-Knoten weiter. Ist auch innerhalb des Backbone das Ziel unbekannt, so verlässt die Suche den OSPF-Graph.

3.5 Sicherheit

Um die Echtheit eines LSAdv zu gewähren, unterstützt OSPF **Authentifizierung**. Dadurch wird die Herkunft eines Paketes sichergestellt. Es stehen zwei Möglichkeiten zur Auswahl. Klartextpassworte und MD5-Hashs. Beiden gemein ist, dass alle Knoten über ein gemeinsames Passwort verfügen. Im Fall des Klartextpasswort wird dieses im Header der Nachricht versendet und kann - sofern nicht andere Verschlüsselungsmaßnahmen ergriffen worden - von jedermann mitgelesen werden. Als Alternative existiert das MD5-Verfahren. Hier wird aus Nachricht und Passwort eine Prüfsumme gebildet die zusammen mit dem Paket versendet werden. Die Empfänger können aus der empfangenen Nachricht und dem Passwort die Prüfsumme verifizieren.

4 Zusammenfassung

Das Routing Information Protocol ist ein typischer Vertreter der Distanzvektorprotokolle. Es speichert nur einen Teil des Netzwerkes und ist dadurch recht einfach zu implementieren. Die Berechnung der Distanzvektoren ist resourcenschonend und damit auch für low-end Router geeignet. Die schlechten Skalierungseigenschaften und die damit einhergehend hohen Konvergenzzeiten können in kleinen Umgebungen vernachlässigt werden. Als einziges Manko bleibt die unzureichenden Authentifizierungseigenschaften. Zwar ist auch hier einen Lösung mittels Absicherung der UDP-Schicht per IPSec (AH Mode - RFC2080) denkbar, würde aber die genannten Vorteile relativieren. Mit zunehmender Rechenleistung und wachsenden Netzen werden Link-State-Protokolle für die breite Masse immer interessanter. Da aber wahrscheinlich die größte Motivation für den Einsatz von Routingprotokollen in Ausfallsicherheit und Redundanz liegt, sind Link-State-Protokolle von Natur aus erste Wahl. Neben den allmächtigen OSPF setzt sich aber zusehens IS-IS wegen seiner Einfachheit durch.

Literatur

[1] RIP. http://tools.ietf.org/html/rfc1058, 2008-01-12, 1988.

[2] OSPF. http://tools.ietf.org/html/rfc2328, 2008-01-12, 1998.

[3] RIPv2. http://tools.ietf.org/html/rfc2453, 2008-01-12, 1998.

[4] R. E. Bellman. *On a Routing Problem, Quarterly of Applied Mathematics.* 1958.

[5] L. R. Ford. *L. R. Ford: Network flow theory.* 1956.

[6] Keith W. Ross James F. Kurose. *Computer Networking, A Top-Down Approach Featuring the Internet.* 1995.

[7] Peter Widmayer Thomas Ottmann. *Algorithmen und Datenstrukturen.* 1995.