Kurzbericht Projekt 11

OSPF Simulation in Haskell

Reto Lehnherr, Gabriel Zimmerli

Abstract
TODO: Zusammenfassung, schreiben wenn alles bereit ist

Idee des Projekts

Wir möchten einen Teil des OSPF-Protokolls implementieren. Die Idee ist, dass unser Programm ein Router im Netz simuliert. Anhand von einer Tabelle sollen die Nachbarschaften zu anderen Routern deklariert sein.

Von diesen Nachbarn werden dann Link-State-Updates empfangen. Da unser Programm lediglich einen Router darstellt, werden dies Updates Files auf dem Dateisystem sein. Diese Files müssen ausgewertet werden um anschliessend eine Topologie Tabelle erstellen zu können. Danach wird anhand dieser Topologie Tabelle mit Hilfe des Dijkstra Algorithmus der Shortest Path Tree und damit schlussendlich die Routing-Tabelle erstellt.

Als Input haben wir ein Text-File mit einer Nachbarschaftstabelle, wie sie der Router nach Erhalt der OSPF-Hello Pakete erstellt hat. Als weiterer Input dient ein weiteres Text-File, welches Link State Updates aufführt, welche von jedem Router aus der Nachbarschaft geschickt werden. Mit diesen Informationen wird die Routing Tabelle berechnet, welche in ein File geschrieben wird.
Der Schwerpunkt liegt beim Parsen der Files, dem Auswerten der Link State Updates und der Implementierung des Dijkstra-Algorithmus.

Hintergrund

Das Open Shortest Path First (OSPF) Protokoll ist ein Link-State-Routing-Protokoll für IP-Netzwerke und basiert auf den Dijkstra-Algorithmus. Das OSPF Protokoll sammelt Link-State Informationen von den verfügbareren Routern und erstellt so eine Topologie des Netzwerks. Das Routing Protokoll berechnet den kürzesten Pfad anhand der «Link-Kosten» welche zu jedem Knoten im Netzwerk gespeichert wurden.

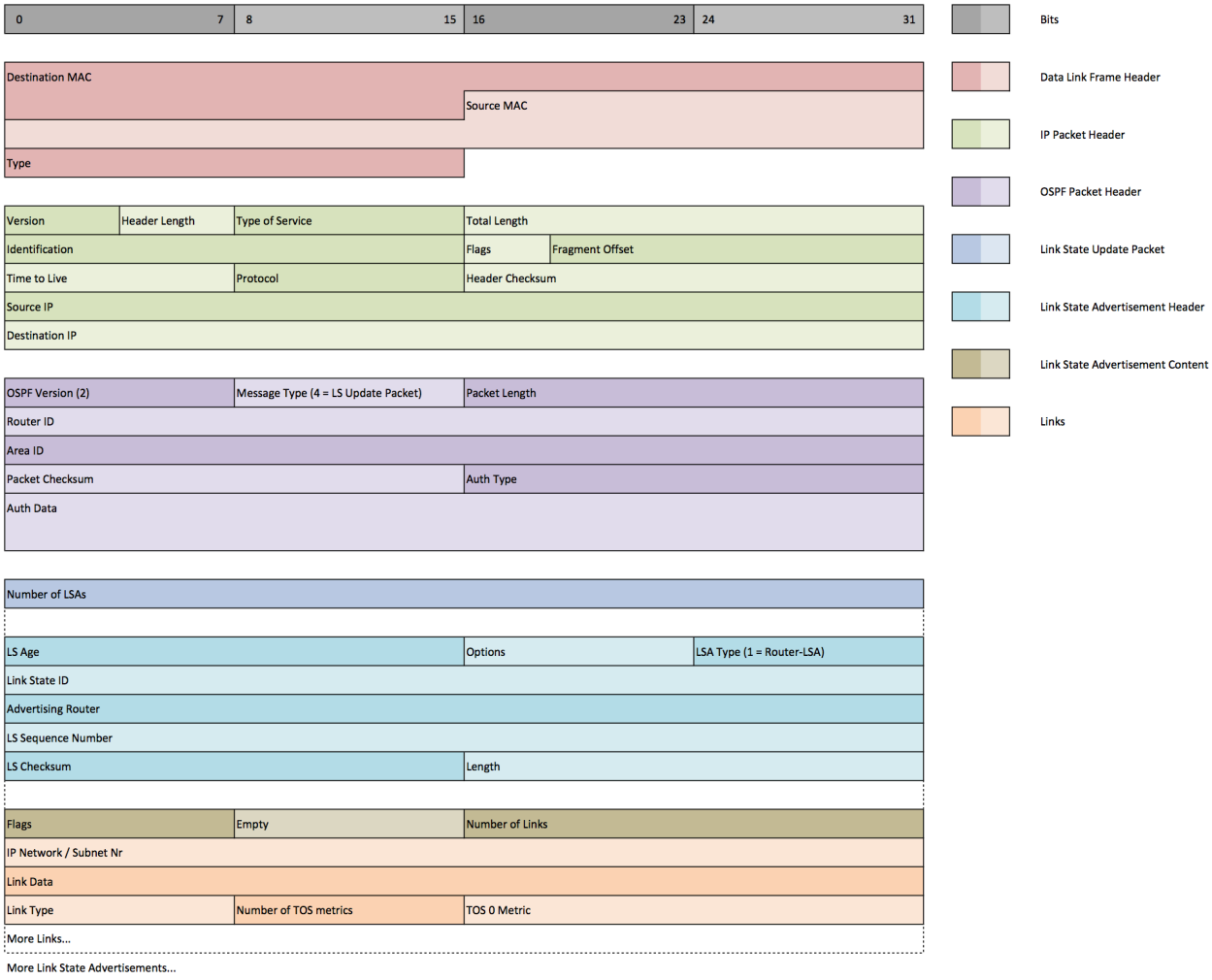
Wichtige Begriffe (TODO)
Hello Paket
Link State Updates
Routing Tabelle

Erstellung Nachbarschaftstabelle
Ein Router der OSPF ausführt erstellt mit Hilfe des Hello Protokolles eine Nachbarschaftstabelle. Diese könnte folgendermassen aussehen:

Address	Interface	State	ID	Priority	Dead
10.1.34.2	so-0/0/0.0	Full	10.0.0.4	128	36
10.1.23.1	so-0/0/1.0	Full	10.0.0.2	128	38
10.1.36.2	so-0/0/3.0	Full	10.0.0.6	128	33

Um das ganze in wenig zu vereinfachen, werden wir diesen Schritt weg lassen und gehen davon aus, dass die Nachbarschaften bereits aufgebaut wurden. Unser Programm hat (ein Router) liest eine solche Nachbarschaftstabelle aus einer Textdatei ein.

Erstellung Topology Table
Nachdem die Nachbarschaften ermittelt wurden erstellt der Router anhand von Link State Updates eine Topology Table (oder auch Link State Database). Dazu müssen erst mal die Link State Updates ausgewertet werden. In unserem Projekt wird ein solches Update als Datei im Filesystem repräsentiert. Jede solche Datei repräsentiert wiederum ein Link State Update eines benachbarten Routers.
Eine solche Datei beinhaltet eine Kette von Hexadezimalen Werten und weist folgende Struktur auf:



Nachdem die Link State Updates eingelesen und ausgewertet wurden, wird daraus die Topologie Tabelle erstellt. Diese stellt dar, welche Router miteinander verbunden sind und wie hoch die Kosten dieser Verbindungen sind.

Erstellung Routing Table

Der Router erstellt anhand der Topology Table eine Routing Table. Er berechnet jeweils den kürzesten Pfad zu einer Adresse basierend auf der konfigurierten Metrik, im Falle von OSPF ist dies die Bandbreite einer Verbindung. Um den kürzesten Pfad zu berechnen, erstellen wir aus den Daten einen gewichteten Graphen, wobei das Gewicht einer Kante aus der «cost metric», der Bandbreite einer Verbindung, bestimmt wird. In diesem Graphen berechnen wir für jeden Knoten den kürzesten Pfad, mithilfe des Dijkstra-Algorithmus.

Metric

Wir verwenden dieselbe Metric, welche laut Wikipedia auch von Cisco verwendet wird. Eine Metrik wird definiert durch 10^8/Bandbreite.

Bandbreite	Cost Metric
100 Mbit/s	1
Gigabit-Ethernet (1Gbit/s)	0.1
10 Mbit/s	10

Aufbau Routingtabelle

Gateway of last resort is not set

```
141.108.0.0/16 is variably subnetted, 8 subnets, 3 masks
O 141.108.1.128/25 [110/65] via 141.108.10.10, 00:15:28, Serial0/0
O 141.108.9.128/25 [110/129] via 141.108.10.10, 00:15:28, Serial0/0
O 141.108.1.0/25 [110/65] via 141.108.10.10, 00:15:28, Serial0/0
C 141.108.10.8/30 is directly connected, Serial0/0
O 141.108.9.0/25 [110/129] via 141.108.10.10, 00:15:28, Serial0/0
O IA 141.108.10.0/30 [110/192] via 141.108.10.10, 00:15:28, Serial0/0
O 141.108.12.0/24 [110/129] via 141.108.10.10, 00:15:28, Serial0/0
O 141.108.10.4/30 [110/128] via 141.108.10.10, 00:15:29, Serial0/0
131.108.0.0/16 is variably subnetted, 9 subnets, 4 masks
C 131.108.4.128/25 is directly connected, Loopback1
O 131.108.5.32/27 [110/1010] via 131.108.1.2, 00:16:04, Ethernet0/0
O 131.108.33.0/24 [110/74] via 141.108.10.10, 00:15:29, Serial0/0
O 131.108.6.1/32 [110/11] via 131.108.1.2, 00:16:04, Ethernet0/0
C 131.108.5.0/27 is directly connected, Loopback2
O 131.108.6.2/32 [110/11] via 131.108.1.2, 00:16:06, Ethernet0/0
C 131.108.4.0/25 is directly connected, Loopback0
C 131.108.1.0/24 is directly connected, Ethernet0/0
O 131.108.26.0/24 [110/138] via 141.108.10.10, 00:15:31, Serial0/0
```

Listing 1 Beispiel einer Routing Tabelle. Quelle:
http://www.ciscopress.com/articles/article.asp?p=26919&seqNum=7

Eintrag in einer Routing Table

```
O 141.108.1.128/25 [110/65] via 141.108.10.10, 00:15:28, Serial0/0
```

O	steht für das Routing-Protokoll, in unserem Fall immer O = OSPF
141.108.1.128/25	Ziel-Adresse
[110/65]	[Administrative Distanz/Metrik] Die Administrative Distanz des Protokolls ist bei OSPF 110

	Die Metrik wird anhand der link cost berechnet und gibt das Gewicht eines Pfades an
`via 141.108.10.10`	Gibt an, ob die Route direkt oder über einen anderen Router erreichbar ist.
`00:15:28`	Dead-Time
`Serial0/0`	Interface Type

Einschränkungen in unserem Projekt:

- Die Routing Table berücksichtigt nur OSPF-Routen
- Die Prefix-Länge einer Adresse wird nicht ausgewertet
- Wir beschränken uns auf Router in einer Area ohne Verbindungen in eine andere
- Wir verzichten auf die Bestimmung eines Designated und Backup Designated Routers
- Bei den Link State Advertisement Typen (LSA) beschränken wir uns auf LSA1

Source Code

TODO

Quellen:

- http://en.wikipedia.org/wiki/Open_Shortest_Path_First
- http://de.wikipedia.org/wiki/Open_Shortest_Path_First
- http://en.wikipedia.org/wiki/Link-state_advertisement
- http://de.wikipedia.org/wiki/Link_State_Advertisement
- Skript «CCNA2 Routing-Konzepte und -Protokolle», Peter Gysel, IMVS FHNW
- Wireshark Beispiel-Capture
- http://www.ciscopress.com/articles/article.asp?p=26919&seqNum=7
- The Routing Table v1.12 von Aaron Balchunas, http://www.routeralley.com/ra/docs/routing_table.pdf
- http://www.routeralley.com/ra/docs/ospf.pdf
- http://www.juniper.net/techpubs/en_US/junos9.6/information-products/topic-collections/nog-baseline/ospf-neighbors-introduction.html
- http://www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a0080094e9e.shtml
- http://www.tcpipguide.com/free/t_OSPFBasicTopologyandtheLinkStateDatabase.htm