TT1 Praktikum 1 & 2 : Ausarbeitung

Carsten Noetzel, Armin Steudte

09.11.2011

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Projektschritt 1 - Aufbau

Im ersten Praktikumstermin wurden die Router konfiguriert und mit den nötigen Protokollen versehen. Die Versuchsdurchführung im zweiten Praktikumstermin wurde in Zusammenarbeit mit André Harms und Oliver Steenbruck durchgeführt und die Ergebnisse dokumentiert.
Der unter Abbildung 1 dargestellte Versuchsaufbau liegt den Versuchen zu Grunde. Hierbei wurde im Wechsel ein Router der Gruppe Noetzel, Steudte und der Gruppe Harms, Steenbruck aufgestellt und die Messungen zwischen den beiden äußersten Routern durchgeführt. Die Anzahl der Hops wurde mittels Tracert-Befehlen und bei OLSR über die Weboberfläche ermittelt. Dabei wurde angestrebt, dass die beiden äußersten Router über die beiden Router in der Mitte kommunizieren, was sich in den Versuchen durch geschickte Positionierung der Router auch als erfüllbar erwies.

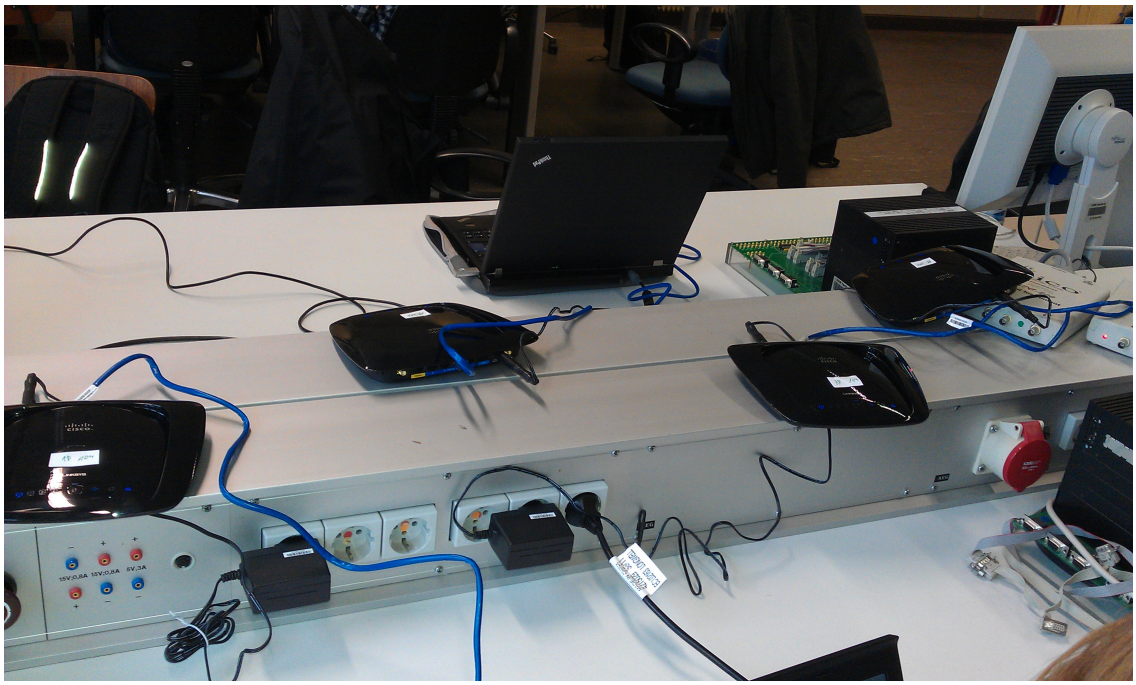

Abbildung 1: Versuchsaufbau mit 4 Routern

2 Projektschritt 2 - Vergleich Babel und OLSR

2.1 Babel

2.1.1 Informationsaustausch

Jeder Router A hält beim Babel-Protokoll die Kosten zu einem Nachbarrouter B in der Form C(A, B) vor. Die Metrik einer Route bestimmt sich über die Summierung aller Kosten zwischen den Knoten die auf der Route liegen. Das Ziel des Algorithmus liegt darin für jede Quelle S einen Baum der Routen, mit den niedrigsten Metriken zu S zu berechnen.

Babel nutzt *Hello-Messages* zur Durchführung des *Neighbour-Discovery-Processes*. Hello-Messages werden periodisch über alle Interfaces des Knoten gesendet um so die Nachbarknoten zu entdecken.
Zusätzlich werden Hello-Messages in Verbindung mit den *IHY-Messages* (I Heard You Messages) genutzt, um die *Bidirectional Reachability* und die Metrik zwischen Sender- und Empfängerknoten zu ermitteln (vgl. Abbildung 2). Der Empfänger antwortet auf Hello-Messages mit den IHY-Messages. Diese erhalten die, mit Hilfe der Hello-Messages ermittelten, Laufzeitzeitverzögerungen aus Sicht des Senders der Hello-Messages. Mit Hilfe der Laufzeitverzögerungen findet dann die Bewertung der Qualität der Links, sowohl in Sende- als auch Empfangsrichtung statt.

Was aus Abbildung 2 hervorgeht und uns so vorher nicht bewusst war, ist dass die Knoten zum Austausch von Nachrichten unter Babel *IPv6* nutzen. Die ausgetauschten Nachrichten enthalten dabei sowohl auf IPv6- als auch IPv4-Adressen und werden an eine IPv6-Multicast-Group gesendet.

```
⊞ Frame 12: 90 bytes on wire (720 bits), 90 bytes captured (720 bits)
⊞ Ethernet II, Src: Cisco-Li_d0:8f:81 (c0:c1:c0:d0:8f:81), Dst: IPv6mcast_00:01:00:06 (33:33:00:01:00:06)
⊞ Internet Protocol Version 6, Src: fe80::c2c1:c0ff:fed0:8f81 (fe80::c2c1:c0ff:fed0:8f81), Dst: ff02::1:6 (ff02::1:6)
⊞ User Datagram Protocol, Src Port: babel (6696), Dst Port: babel (6696)
⊟ Babel Routing Protocol
    Magic: 42
    Version: 2
    Body Length: 24
  ⊟ Message hello (4)
      Message Type: hello (4)
      Message Length: 6
      Seqno: 0x49dd
      Interval: 400
  ⊟ Message ihu (5)
      Message Type: ihu (5)
      Message Length: 14
      Rxcost: 0x0111
      Interval: 1200
    ⊞ Address: fe80::c2c1:c0ff:fed0:8ed0
```

Abbildung 2: Beispiel für eine Babel-Hello-Message

2.1.2 Mesh - Konfiguration

Die lokalen Mesh-Konfigurationen werden über einen verteilten Bellman-Ford Algorithmus berechnetet.
Hierzu hält jeder Router A zwei Werte vor, zum einen die geschätzte Distanz zu S (bezeichnet als D(A)) und den Next-Hop-Router zu S (bezeichnet als NH(A)). Zu Beginn des Informationsaustausches ist D(A)=unendlich und NH(A) ist nicht definiert.
Periodisch sendet jeder Knoten B ein Update seiner Routen an alle seine Nachbarn mit dem Inhalt D(B), welcher die Distanz zu S angibt. Erhält ein Nachbar A von B ein Update, so prüft A ob B für S als Next-Hop-Router eingetragen ist. Ist dies der Fall wird als Distanz D(A) die Summe aus C(A,B)(= Kosten von A nach B) und D(B)(= Kosten von B zu S) gesetzt. Damit ist im Knoten A die Metrik von A nach S über B aktualisiert.
Im Fall, dass B nicht als Next-Hop-Router für S auf A eingetragen ist, vergleicht A die Summe aus C(A,B)(= Kosten von A nach B) und D(B)(= Kosten von B zu S) mit dem gegenwärtigen Wert von D(A). Ist C(A,B)+D(B) kleiner als D(A) ist die angebotene Route besser als die bisher eingetragene und NH(A) wird auf B und D(A) auf C(A,B)+D(B) gesetzt.
Im Pseudocode sieht dies in etwa so aus:

```
1 receiveRouteupdate(D(B)) from B for S
2 If NH(A) = B Then
3         D(A) = C(A,B)+D(B)
4 Else
5         If C(A,B)+D(B) < D(A) Then
6                 NH(A) = B
7                 D(A) = C(A,B)+D(B)
8         EndIf
9 EndIf
```

Durch den Austausch der Updates wird Mesh-Konfiguration festgelegt und die Knoten wissen an welchen Knoten sie Pakete weiterleiten müssen, um ein bestimmtes Ziel mit möglichst geringen geschätzten Gesamtkosten zu erreichen.

2.1.3 Loop-Verhinderungsstrategie

Da es beim Bellman-Ford Algorithmus nach einer Topologieänderung und den damit verbundenen Updates zum *Count-to-infinity*-Problem kommen kann, benötigt Babel eine Loop-Verhinderungs-Strategie.
Hierzu werden in Babel die sogenannten *Feasibility Conditions* eingeführt. Dabei werden, erhalte-

ne Routenaktualisierungen verworfen, wenn diese nicht beweisen können, dass die Annahme des Updates zu keinem Loop führen. Router A hält dazu eine *Feasibility Distance* (bezeichnet als FD(A)) vor, welche als Wert die kleinste Distanz zu S enthält, die A jemals angeboten wurde. Ein Update von einem Nachbarn B ist zulässig (feasible), wenn die Metrik D(B) (= Kosten von B zu S) kleiner als As FD(A) ist.
Das diese Bedingung weniger restriktiv als die *DSDV-Feasibility* ist, macht das Beispiel in Abbildung 3 deutlich. A kommt über B zu S, womit D(A)=FD(A)=2 ist. Da D(C)=1 und damit kleiner als FD(A) ist, ist eine alternative Route über C für A zulässig, auch wenn die Metrik C(A,C)+D(C) mit 5 schlechter als die aktuelle Route ist. Für den Fall das die Route über B abbricht kann so die Alternativroute genutzt werden.
Die *DSDV-Feasibility* besagt, dass ein Update nur dann angenommen werden darf, wenn C(A,C)+D(C) kleiner oder gleich D(A) ist, wodurch die Alternativroute über C nicht zulässig wäre, da die DSDV-Bedingung nicht erfüllt ist (4+2 <= 2).
Um zu zeigen, dass die *Feasibility Condition* trotzdem noch Loop-Freiheit garantiert, sei zu beachten, dass wenn A ein Update von B akzeptiert D(B) nicht kleiner als FD(B) sein kann. Zudem gilt FD(B) ist kleiner als FD(A) da das Update für A zulässig ist und dieser es annimmt. Da diese Eigenschaft weiterhin erhalten bleibt, wenn A Updates versendet, ist sichergestellt, dass die Route keine Loops enthält.

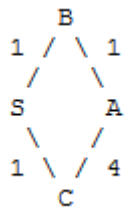

Abbildung 3: Beispiel Feasabiliy Condition aus RFC6126

Um zu verhindern, dass ein Router aufgrund fehlender zulässiger Routen ausgeschlossen wird, werden in Babel die Routen durchnummeriert. Ein Update von B an A ist damit zulässig, wenn entweder die *Feasibility Condition* D(B)<FD(A) erfüllt ist und die das Update die gleiche Sequenznummer besitzt oder aber wenn die Sequenznummer des Updates höher ist.

2.2 OLSR

Bei *OLSR* handelt es sich im Gegensatz zu Babel, wie der Name schon sagt, um einen Vertreter aus der Familie der Link-State-Routing Protokolle. Dabei enthält OLSR gegenüber anderen Vertretern die Optimierung, dass die Knoten ihre Informationen nur an ihre *Multipoint Relays (MPR)* senden müssen und diese das Verbreiten der Link-State Informationen übernehmen. Somit ermöglicht OLSR ein effizientes *Flooding* der Informationen.
OLSR ist proaktiv, wodurch Routen unmittelbar vorhanden sind wenn diese gebraucht werden.

2.2.1 Informationsaustausch

Wie auch in Babel tauschen Nachbarknoten bei OLSR *Hello-Messages* aus. Die Nachrichten haben dabei drei Aufgaben:

- Erkennen von Links auf den Interfaces
- Neighbor Detection

- Bekanntgabe von MPRs

Ein Beispiel für eine Hello-Message ist in Abbildung 4 zu sehen. Zusätzlich werden auch noch

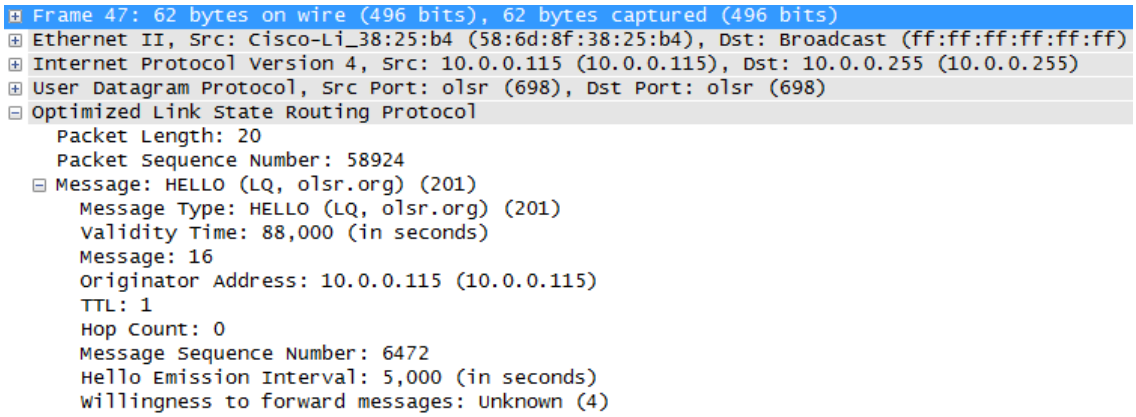

Abbildung 4: Beispiel für eine OLSR-Hello-Message

TC Messages, sogenannte *Topology Control Messages*, über alle OLSR-Interface gesendet (vgl. Abbildung 5). Sie enthalten Informationen über die lokale Topologie der Knoten und dienen zur Ermittlung der globalen Sicht auf das Netzwerk und dem Aufbau des Routing. Sie werden, wie bereits erwähnt, nur durch die MPRs weitergeleitet.
Sollte OLSR auf mehreren Interfaces gleichzeitig laufen werden auch noch *MID Messages (Main*

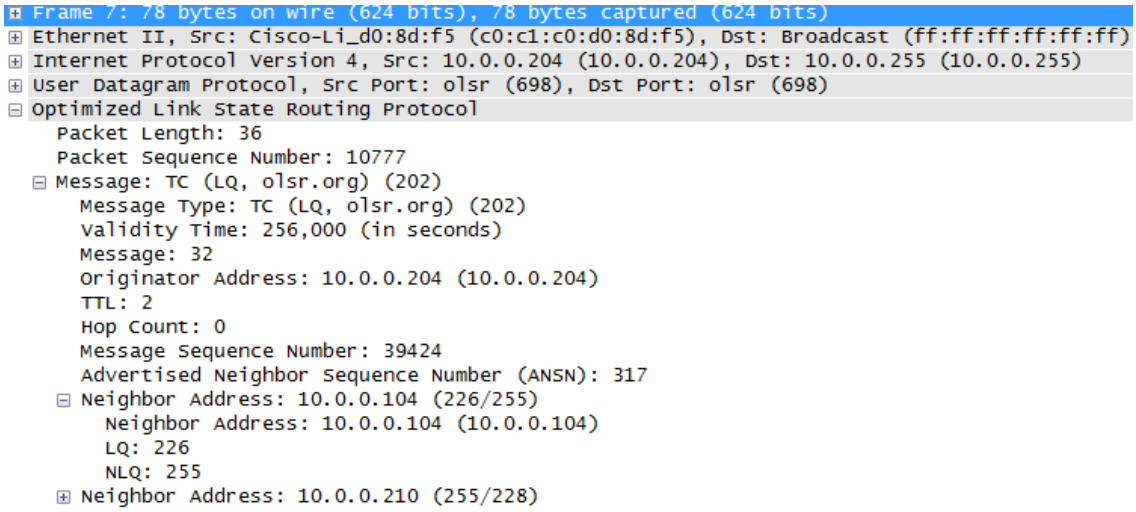

Abbildung 5: Beispiel für eine OLSR-TC-Message

Adress and Multiple Interfaces) gesendet, als zusätzliche Information zum Aufbau des Routing. Da in unserem Fall die Router jeweils nur über die WLAN-Interfaces OLSR ausführten, wurden keine MID Messages generiert.

Bei der Bewertung der Qualität von Links kann auf zwei Methoden zurückgegriffen werden. Standardmäßig wird eine gewisse Anzahl an erfolgreich gesendeten Hello-Messages als Grundlage für die Bewertung eines Links genutzt ob dieser als symmetrisch oder asymmetrisch eingestuft wird. Alternativ wird das *Link Hysteresis*-verfahren genutzt, wobei mit Schwellwerten gearbeitet wird, ab wann ein Link als stabil und damit als symmetrisch gilt. Beide Verfahren können Informationen

aus dem Link-Layer für eine bessere Bestimmung mit einbeziehen. Außerdem existiert noch eine *Link Quality Extension*, die die Hello-Messages nutzt um eine Wahrscheinlichkeit für den Paketverlust auf einem Link zu berechnen. Pakete werde dann in diesem Fall über den Link mit der geringsten Wahrscheinlichkeit des Paketverlustes gesendet.
Wie in Abbildung 5 zu sehen ist, wird zu jedem Nachbarknoten ein Wert für die Link-Qualität in Sende- und Empfangsrichtung in der TC-Message mit gesendet. Diese heißen *LQ* und *NLQ* und weißen darauf in, dass in dem von uns eingesetzten *OLSR-Daemon* die Link Quality Extension aktiv war.

2.2.2 Mesh - Konfiguration

Jeder Knoten im Netzwerk hält eine Routingtabelle mit folgenden Daten vor:

1. R-dest-addr R-next-addr R-dist R-iface-addr
2. R-dest-addr R-next-addr R-dist R-iface-addr
3. ,, ,, ,, ,,

Jeder Eintrag der Liste gibt an, welches Ziel (R-dest-addr) über welchen Next-Hop-Router (R-next-addr) erreichbar ist und welches lokale Interface (R-iface-addr) dafür genutzt werden muss. Ferner wird die Distanz vom lokalen Router bis zum Ziel (R-dist) in der Tabelle vorgehalten.
Die Daten basieren auf Informationen der *local link information dase* und der Topologie, wodurch die Routingtabelle jedes Mal neu berechnet werden muss, wenn sich diese Daten ändern. Für jedes Ziel im Netzwerk zu dem eine Route bekannt ist, wird ein Eintrag in der Tabelle vorgehalten.

2.3 Vergleich Babel und OLSR

3 Projektschritt 3 - Vergleich der Übertragungsqualität