Lehrstuhl für Netzarchitekturen und Netzdienste
Fakultät für Informatik
Technische Universität München

Grundlagen Rechnernetze und Verteilte Systeme

IN0010, SoSe 2019

Übungsblatt 9

1. Juli – 5. Juli 2019

Hinweis: Mit * gekennzeichnete Teilaufgaben sind ohne Lösung vorhergehender Teilaufgaben lösbar.

In dieser Woche wird in den Übungen auch Aufgabe 3 der Midterm besprochen.

Aufgabe 1 Distanz-Vektor-Routing

Gegeben sei die in Abbildung 1 dargestellte Topologie mit den vier Routern A bis D. Die Linkkosten sind jeweils an den Kanten angegeben. Wir notieren die Routingtabellen in Kurzform als Vektor $[(x_A, y_A), \ldots, (x_D, y_D)]$. Die Tupel (x, y) geben dabei die Kosten sowie den Next-Hop zum Ziel an.

Zu Beginn seien die Routingtabellen noch leer, d. h. die Router kennen noch nicht einmal ihre direkten Nachbarn. Dies wird durch die Schreibweise `(/,/)` angedeutet. Sich selbst erreichen die Router natürlich mit Kosten 0.

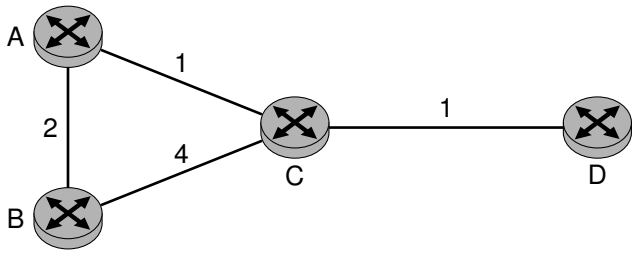

Abbildung 1: Netztopologie (Aufgabe 3)

Die Router beginnen nun damit, in periodischen Zeitabständen ihre Distanz-Vektoren mit ihren direkten Nachbarn auszutauschen. Dabei schickt beispielsweise Router B ein Update an Router C, welches lediglich die Distanz zum jeweiligen Ziel enthält (nicht aber den Next-Hop). Wenn nun Router A ein solches Update von B erhält und darin eine Route zu D finden würde, so wüsste A, dass er D über B erreicht. Die Kosten zu D entsprechen dann den Kosten zu B zuzüglich der Kosten, mit denen B das Ziel erreichen kann.

Im Folgenden wollen wir dieses Verhalten untersuchen. Da das Ergebnis allerdings davon abhängt, in welcher Reihenfolge Updates ausgetauscht werden, treffen wir die idealisierte Annahme, dass alle Router exakt zeitgleich ihre Updates verschicken.

a)* Geben Sie gemäß obiger Tabelle die Routingtabellen aller vier Router in den folgenden Schritten an. Brechen Sie ab, sobald ein konvergenter Zustand erreicht ist.

Schritt	Router A	Router B	Router C	Router D
0	`[ (0,A) (/,/) (/,/) (/,/) ]`	`[ (/,/) (0,B) (/,/) (/,/) ]`	`[ (/,/) (/,/) (0,C) (/,/) ]`	`[ (/,/) (/,/) (/,/) (0,D) ]`
1	`[ (0,A) (2,B) (1,C) (/,/) ]`	`[ (2,A) (0,B) (4,C) (/,/) ]`	`[ (1,A) (4,B) (0,C) (1,D) ]`	`[ (/,/) (/,/) (1,C) (0,D) ]`
2	`[ (0,A) (2,B) (1,C) (2,C) ]`	`[ (2,A) (0,B) (3,A) (5,C) ]`	`[ (1,A) (3,A) (0,C) (1,D) ]`	`[ (2,C) (5,C) (1,C) (0,D) ]`
3	`[ (0,A) (2,B) (1,C) (2,C) ]`	`[ (2,A) (0,B) (3,A) (4,A) ]`	`[ (1,A) (3,A) (0,C) (1,D) ]`	`[ (2,C) (4,C) (1,C) (0,D) ]`

b) Welcher (Graph-)Algorithmus findet hier Verwendung?

Es kommt eine verteilte (dezentrale) Implementierung des Algorithmus von Bellman-Ford zum Einsatz.

Prof. Dr.-Ing. Georg Carle
carle@tum.de
Dr.-Ing. Stephan Günther, Henning Stubbe
grnvs@net.in.tum.de

Nun fällt die Verbindung zwischen den Knoten C und D aus. Die Knoten C und D bemerken dies und setzen die entsprechenden Pfadkosten auf unendlich.

c) Was passiert in den folgenden Schritten, in denen die aktiven Knoten weiter ihre Distanzvektoren austauschen? Geben Sie nach jedem Schritt die Distanztabellen an, bis das weitere Ergebnis klar ist.

Schritt	Router A	Router B	Router C	Router D
4	[(0,A) (2,B) (1,C) (2,C)]	[(2,A) (0,B) (3,A) (4,A)]	[(1,A) (3,A) (0,C) (/,/)]	[(/,/) (/,/) (/,/) (0,D)]
5	[(0,A) (2,B) (1,C) (6,B)]	[(2,A) (0,B) (3,A) (4,A)]	[(1,A) (3,A) (0,C) (3,A)]	[(/,/) (/,/) (/,/) (0,D)]
6	[(0,A) (2,B) (1,C) (4,C)]	[(2,A) (0,B) (3,A) (7,C)]	[(1,A) (3,A) (0,C) (7,A)]	[(/,/) (/,/) (/,/) (0,D)]
7	[(0,A) (2,B) (1,C) (8,C)]	[(2,A) (0,B) (3,A) (6,A)]	[(1,A) (3,A) (0,C) (5,A)]	[(/,/) (/,/) (/,/) (0,D)]
8	[(0,A) (2,B) (1,C) (6,C)]	[(2,A) (0,B) (3,A) (9,C)]	[(1,A) (3,A) (0,C) (9,A)]	[(/,/) (/,/) (/,/) (0,D)]
9	[(0,A) (2,B) (1,C) (10,C)]	[(2,A) (0,B) (3,A) (8,A)]	[(1,A) (3,A) (0,C) (7,A)]	[(/,/) (/,/) (/,/) (0,D)]
10	[(0,A) (2,B) (1,C) (8,C)]	[(2,A) (0,B) (3,A) (11,C)]	[(1,A) (3,A) (0,C) (11,A)]	[(/,/) (/,/) (/,/) (0,D)]
⋮	⋮	⋮	⋮	⋮

d)* In der Vorlesung wurden **Split Horizon**, **Triggered Updates** und **Path Vector** als mögliche Gegenmaßnahmen für das Count-to-Infinity-Problem genannt. Erläutern Sie in der Gruppe die Funktionsweise dieser Verfahren.

- **Split Horizon**
  „Bewerbe eine Route nicht über das Interface, über das die Route ursprünglich gelernt wurde.“ Im konkreten Fall würden also Router A und B keine Route zu D an C schicken. Durch die ringförmige Topologie wird das Problem dadurch jedoch noch nicht behoben.
- **Triggered Update**
  Anstelle Routing-Updates nur in periodischen Zeitabständen zu versenden (bei RIP standardmäßig 30 s), werden Updates sofort geschickt, wenn Topologieänderungen erkannt werden. Dies löst das Count-to-Infinity-Problem zwar nicht, beschleunigt den Vorgang aber. Das Problem besteht darin, dass kurzzeitig viel Datenverkehr durch Routingupdates verursacht wird. Triggered Updates werden von den meisten Routingprotokollen unterstützt (RIP erst ab Version 2).
- **Path Vector**
  Es wäre möglich, dass sich jeder Router bei einem Update merkt, woher das Update kam und diese Information bei Routing-Updates mit einschließt. Auf diese Weise könnte jeder Router den vollständigen Pfad nachvollziehen, den ein Paket zum jeweiligen Ziel nehmen wird. Entdeckt ein Router sich selbst in diesem Pfad, so weiß er, dass eine Schleife vorhanden ist. Er kann das Update in diesem Fall verwerfen. Dieses Verfahren wird von BGP eingesetzt.

Aufgabe 2 Subnetting (Hausaufgabe)

Der TUMexam AG werden die Adressbereiche 131.159.32.0/22 und 131.159.36.0/24 zugewiesen. Für die Aufteilung dieses Adressbereichs ist die TUMexam AG selbst verantwortlich. Nach einer sorgfältigen Bedarfsanalyse ergeben sich die folgenden Anforderungen an die Subnetze und die Mindestanzahl **nutzbarer** IP-Adressen:

Subnetz	NET 1	NET 2	NET 3	NET 4	NET 5
IPs	300	300	15	40	4

Bei der Erhebung dieser Zahlen wurde die an das jeweilige Router-Interface zu vergebende IP-Adresse bereits berücksichtigt.

a)* Geben Sie jeweils die erste und letzte IP-Adresse der beiden vergebenen Adressbereiche an.

- 131.159.32.0/22:
  Erste IP: 131.159.32.0 (Netzadresse)
  Letzte IP: 131.159.35.255 (Broadcast-Adresse)
- 131.159.36.0/24:
  Erste IP: 131.159.36.0 (Netzadresse)
  Letzte IP. 131.159.36.255 (Broadcast-Adresse)

b) Wie viele IP-Adressen stehen der TUMexam AG insgesamt zur Verfügung? Können alle davon zur Adressierung von Hosts verwendet werden?

- 131.159.32.0/22: $2^{32-22} = 2^{10} = 1024$ Adressen
- 131.159.36.0/24: $2^{32-24} = 2^{8} = 256$ Adressen

Insgesamt stehen also $1024 + 256 = 1280$ Adressen zur Verfügung. Allerdings sind die erste (Netzadresse) und letzte Adresse (Broadcast-Adresse) eines jeden Netzes nicht zur Adressierung von Hosts nutzbar. Es stehen also zunächst maximal $1022 + 254 = 1276$ Adressen zur Hostadressierung zur Verfügung.

c)* Ist es möglich, den von den beiden Adressblöcken gebildeten Adressbereich in einem einzigen Subnetz zusammenzufassen?

Nein. Die Subnetze sind nicht gleich groß (/22 und /24) und können damit keinesfalls zusammengefasst werden, da das nächst größere Subnetz mit einem /21 Präfix in jedem Fall weitere Netze enthalten würde.

(Ein einzelnes Subnetz hat als Größe immer eine Zweierpotenz, wir bräuchten hier aber eines mit $1024 + 256 = 1280$ Adressen.)

Achtung: Das obige Kriterium ist nur notwendig, nicht hinreichend! Zwei gleich große Subnetze können auch nur dann zusammengefasst werden, wenn sie aufeinander folgen **und** sich im nächst größeren Subnetz zusammen fassen lassen. (Letzteres Kriterium ist gleichbedeutend mit einem gemeinsamen Vaterknoten der beiden Subnetze, wenn man sich den Adressraum als Binärbaum vorstellt.)

d)* Teilen Sie nun die beiden Adressbereiche gemäß der Bedarfsanalyse auf, so dass Subnetze der passenden Größe entstehen. Gehen Sie mit den Adressen so sparsam wie möglich um. Es soll am Ende ein möglichst großer zusammenhängender Adressbereich für zukünftige Nutzung frei bleiben. Für jedes Subnetz ist anzugeben:

- die Größe des Subnetzes
- die Anzahl nutzbarer Adressen
- das Subnetz in Präfixschreibweise
- die Subnetzmaske in Dotted-Decimal-Notation
- die Netz- und Broadcastadresse

Um die Vorgaben zu erfüllen, müssen wir die Subnetze gemäß ihrer Größe in absteigender Reihenfolge bearbeiten. Andernfalls könnten wir die folgende Situation erhalten:

- An Netz 3 wird der Adressbereich 131.159.36.0/27 vergeben.
- Vergibt man nun aber an Netz 4 den Bereich 131.159.36.32/26, macht man einen Fehler. Um dies zu verstehen, muss man sich die Binärschreibweise der Netzadresse und Subnetz-Maske ansehen:
  `131.159. 36.0010 0000` (IP)
  `255.255.255.1100 0000` (Subnetz-Maske)
  Eine UND-Verknüpfung beider Zeilen ergibt, dass die IP-Adresse 131.159.36.32 in das Subnetz 131.159.36.0/26 fällt!
- Wir müssten also den Bereich 131.159.36.64/26 an Netz 4 vergeben. Dann allerdings entstünde eine Lücke zwischen Netz 3 und Netz 4.

- Vergibt man die Adressen gemäß der Größe der Subnetze in absteigender Reihenfolge, umgeht man das Problem. Dieses Vorgehen könnte natürlich wieder anderen Kriterien widersprechen – beispielsweise der Vergabe zusammenhängender Adressblöcke an einzelne Niederlassungen.

Subnetz	NET 1	NET 2	NET 3	NET 4	NET 5
Bedarf	300	300	15	40	4
Größe	512	512	32	64	8
Nutzbar	510	510	30	62	6
Präfixnotation	131.159.32.0/23	131.159.34.0/23	131.159.36.64/27	131.159.36.0/26	131.159.36.96/29
Subnetzmaske	255.255.254.0	255.255.254.0	255.255.255.224	255.255.255.192	255.255.255.248
Netzadresse	131.159.32.0	131.159.34.0	131.159.36.64	131.159.36.0	131.159.36.96
Broadcast	131.159.33.255	131.159.35.255	131.159.36.95	131.159.36.63	131.159.36.103