Lehrstuhl für Netzarchitekturen und Netzdienste
Fakultät für Informatik
Technische Universität München

Grundlagen Rechnernetze und Verteilte Systeme (IN0010)

Übungsblatt 9

21. Juni – 25. Juni 2021

Aufgabe 1 Statisches Routing

Wir betrachten die Netztopologie des Unternehmens *TUMexam AG*, welche in Abbildung 1.1 dargestellt ist. Es soll die Erreichbarkeit der Subnetze NET1-3 untereinander sowie mit dem Internet sichergestellt werden. Die Router R1 und R2 sollen jeweils die höchste nutzbare IP-Adresse in den jeweiligen Subnetzen erhalten. Zur Verbindung zwischen den Routern stehen Transportnetze mit jeweils nur zwei nutzbaren Adressen zur Verfügung. Der Router mit dem lexikographisch kleineren Namen (z. B. R1 $\prec$ R2) soll hier die niedrigere IP-Adresse erhalten.
Der Gateway der *TUMexam AG* sei über sein öffentliches Interface `ppp0` mit dem Internet verbunden. Sein Default Gateway sei 93.221.23.1.

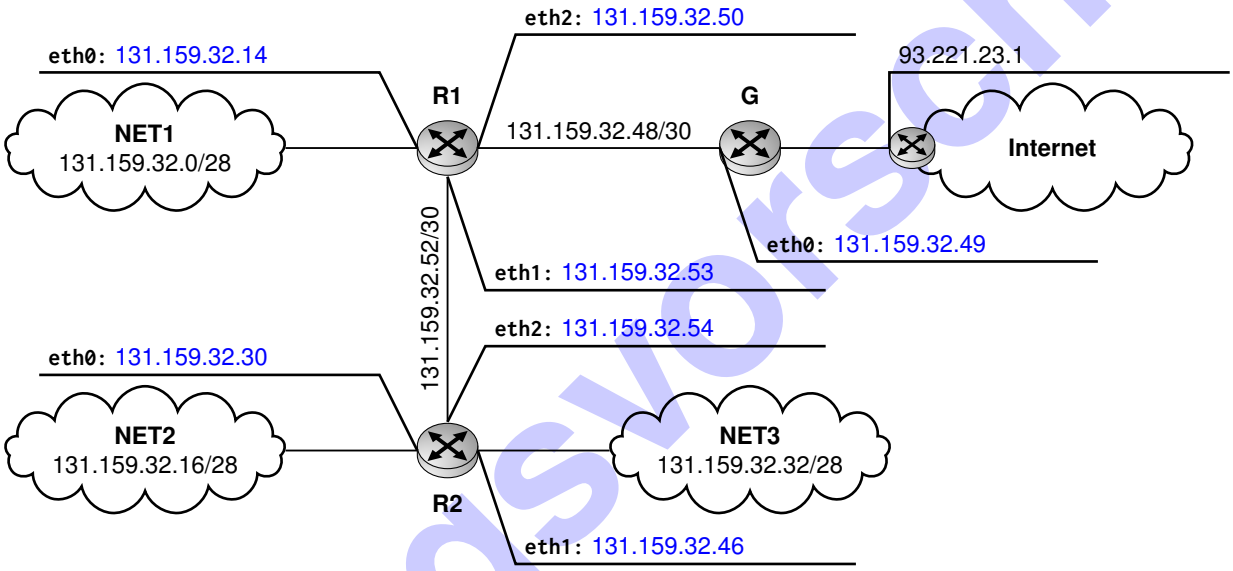

Abbildung 1.1: Netztopologie und IPv4-Adressierung

a)* Weisen Sie jedem Interface der Router R1, R2 und G jeweils eine IPv4-Adresse zu (Router G nur Interface `eth0`). Tragen Sie die Adressen direkt in Abbildung 1.1 ein.

Die Routingtabelle von R2 sei wie folgt gegeben:

Destination	Next Hop	Iface
131.159.32.52/30	0.0.0.0	eth2
131.159.32.16/28	0.0.0.0	eth0
131.159.32.32/28	0.0.0.0	eth1
0.0.0.0/0	131.159.32.53	eth2

Tabelle 1.1: Routing-Tabelle von R2

Der Eintrag `0.0.0.0` in der Spalte „Next Hop“ bedeutet, dass kein Gateway benötigt wird (Netz ist direkt angeschlossen). Die letzte Zeile ist der Eintrag für den sog. *Default-Gateway*. Dorthin werden Pakete an all diejenigen Netze weitergeleitet, für die keine bessere Route bekannt ist.

b) Geben Sie die Routingtabellen der Router R1 und G an. Fassen Sie dabei einzelne Routen soweit möglich zusammen und sortieren Sie die Einträge absteigend in der Länge des Präfixes.

Destination	Next Hop	Iface
131.159.32.48/30	0.0.0.0	eth2
131.159.32.52/30	0.0.0.0	eth1
131.159.32.0/28	0.0.0.0	eth0
131.159.32.16/28	131.159.32.54	eth1
131.159.32.32/28	131.159.32.54	eth1
0.0.0.0/0	131.159.32.49	eth2

Routing-Tabelle von R1

Destination	Next Hop	Iface
131.159.32.48/30	0.0.0.0	eth0
131.159.32.52/30	131.159.32.50	eth0
131.159.32.32/28	131.159.32.50	eth0
131.159.32.0/27	131.159.32.50	eth0
0.0.0.0/0	93.221.23.1	ppp0

Routing-Tabelle von G

c)* Weswegen benötigt Router G nicht notwendiger Weise eine Route ins Transportnetz 131.159.32.52/30?

Router G benötigt diese Route nur dann, wenn er Ziele innerhalb dieses Transportnetzes erreichen muss. Dies ist aber (gemäß der Aufgabenstellung) nicht erforderlich. Auf die Erreichbarkeit der Subnetze NET2 und NET3 hat dies keinen Einfluss!

Die Leitung der *TUMexam AG* hat 2015 beschlossen, nun endlich mit der Migration auf IPv6 zu beginnen. Die zusätzliche IPv6-Adressierung ist in Abbildung 1.2 dargestellt.

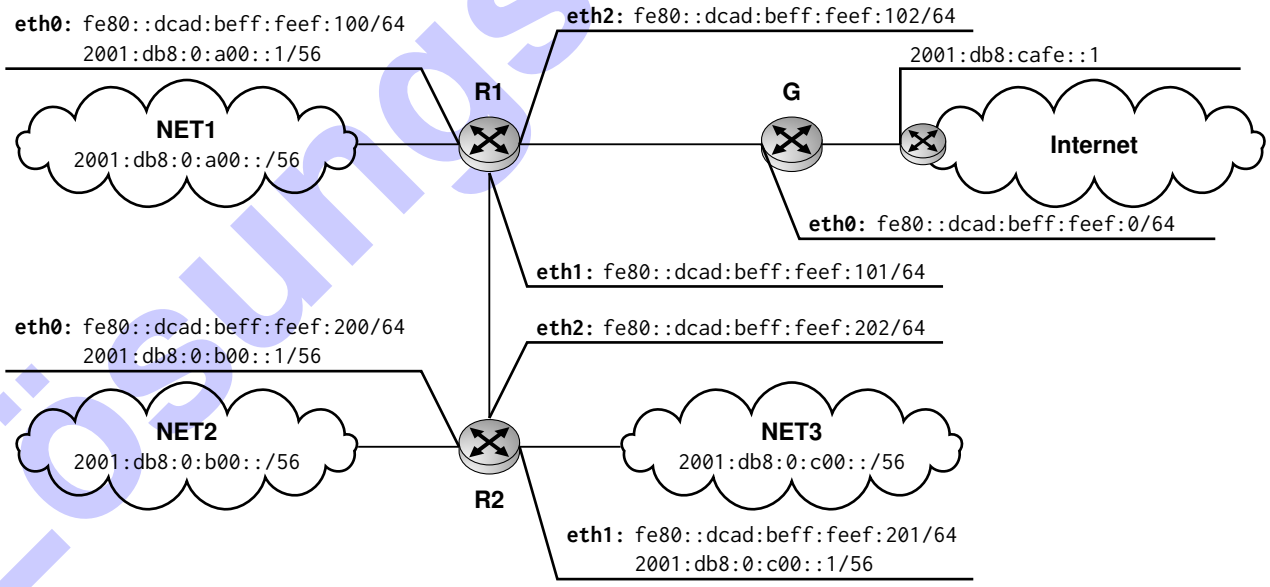

Abbildung 1.2: Netztopologie und IPv6-Adressierung

d)* Was ist der Unterschied zwischen den beiden IPv6-Adressen `fe80::dcad:beff:feef:201/64` und `2001:db8:0:c00::1/56` an Interface `eth1` von R2?

Bei `fe80::dcad:beff:feef:201/64` handelt es sich um eine Link-Local Adresse, die mittels SLAAC zugewiesen wurde. `2001:db8:0:c00::1/56` ist hingegen eine Global-Unique Adresse.

e)* Geben Sie die erste und letzte Adresse des Subnetzes an, zu dem die Adresse `fe80::dcad:beff:feef:201/64` gehört.

`fe80::` – `fe80::ffff:ffff:ffff:ffff`

f) In welchem Subnetz befinden sich demnach die Link-Local Adressen der übrigen Geräte aus Abbildung 1.2?

Alle im selben – nämlich `fe80::/64`.

g) Stellt es ein Problem dar, dass das Subnetz `fe80::/64` offenbar mehrfach vergeben ist?

Nein, da Link-Local Adressen ohnehin nur im lokalen Subnetz (scope link) Gültigkeit haben und niemals geroutet werden.

h) Der Default-Gateway von G sei `2001:db8:cafe::1` und über sein externes Interface `ppp0` erreichbar. Stellen Sie für Router G die IPv6 Routing-Tabelle auf. Fassen Sie dazu wieder Einträge soweit wie möglich zusammen und sortieren Sie die Einträge absteigend in der Länge des Präfixes.

Destination	Next Hop	Iface
`fe80::/64`	`::`	eth0
`2001:db8:0:c00::/56`	`fe80::dcad:beff:feef:102`	eth0
`2001:db8:0:a00::/55`	`fe80::dcad:beff:feef:102`	eth0
`::/0`	`2001:db8:cafe::1`	ppp0

IPv6 Routing-Tabelle von G

Aufgabe 2 Distanz-Vektor-Routing

Gegeben sei die in Abbildung 2.1 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.
Zum Beispiel geht der kürzeste Pfad von A nach B über B mit Kosten 2, von A nach C über C mit Kosten 1 und von A nach D über C mit Kosten 2. Router erreichen sich selbst per Definition mit Kosten 0. Das ergibt für Router A dann die Routingtabelle `[(0,A) (2,B) (1,C) (2,C)]` (die Position innerhalb des Vektors gibt das jeweilige 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.

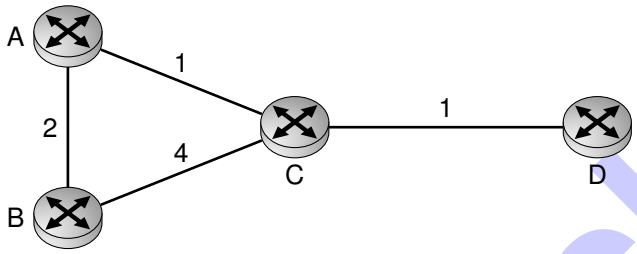

Abbildung 2.1: Netztopologie

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 Definitionen die Routingtabellen aller vier Router in den folgenden Schritten an. Brechen Sie ab, sobald ein konvergenter Zustand erreicht ist.

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

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

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

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