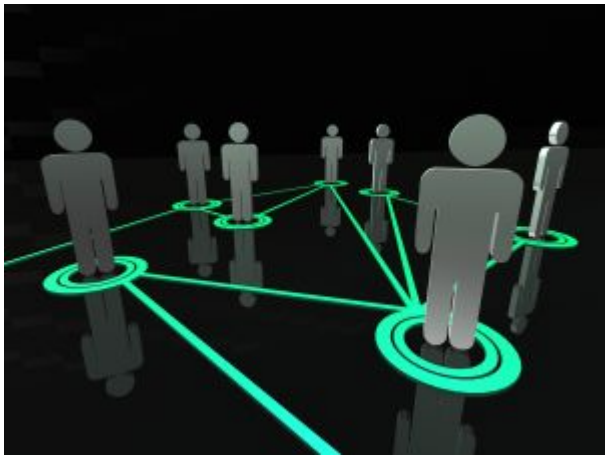
Modul N5 - Routing



http://www.sxc.hu

Zeitrahmen
70 Minuten

Zielgruppe
- Sekundarstufe II

Inhaltliche Voraussetzung
keine, N4 von Vorteil

Lehrziel
Kennen lernen eines Routing-Algorithmus, sowie der Netzwerkschichten

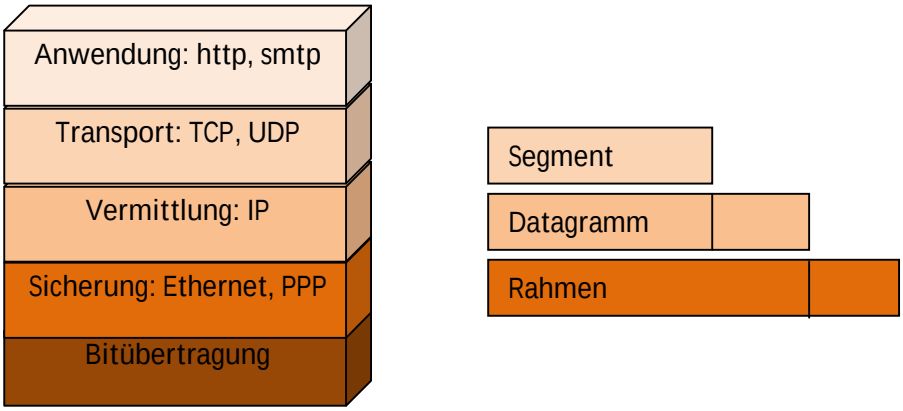
Motivation
Die TN sollen auf spielerische Art und Weise durch die Simulation eines kleinen Netzwerkes einen Routing-Algorithmus kennen lernen. Wie auch im Straßenverkehr geht es darum den kürzesten Weg von einem Punkt zum nächsten zu finden.

Requisiten
Wollfäden, Kärtchen für Kosten der Verbindunglinien sowie für die Benennung der Knotenpunkte: N-AB 5.1 und Vorgefertigte Distanztabellen: N-AB 5.2, N-AB 5.3 Handout

Partizipanden
Gesamte Klasse, für die Animation werden je 3 TN benötigt.

Vorgehensweise

1. Bevor wir beginnen, uns einen Routing-Algorithmus anzusehen sollten wir generell darauf eingehen, wie Computernetzwerke aufgebaut sind. Wie auch bei der Kommunikation zwischen Menschen benötigen wir Vorschriften und Regeln, wenn Computer miteinander kommunizieren. Dafür gibt es Unterschiedliche Netzwerkprotokolle. Jeder Knoten in einem Netzwerk ist in unterschiedliche Schichten unterteilt, wobei jede Schicht eine gewisse Aufgabe erfüllt und bestimmt Protokolle zur Verfügung stellt.
2. Nun können wir zu den Schichten über gehen, die in Netzwerkknoten verwendet werden, allerdings in einer vereinfachten Darstellung (siehe auch AB-N5.3).



Die **Anwendungsschicht** betrifft uns als Benutzer. Sie stellt Protokolle (Kommunikationsregeln) wie etwa HTTP zur Übertragung von Webseiten oder SMTP für E-Mails zur Verfügung.

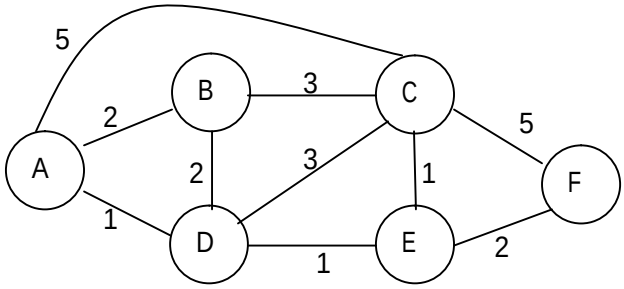
Die **Transportschicht** packt die Nachricht der Anwendungsschicht in Segmente. Sie ist für die Kommunikation zwischen Client- und Serverseite zuständig.

In der **Vermittlungsschicht** werden die Segmente in Datagramme gepackt. D.h. das Segment bekommt noch eine Zusatzinformation (Header) dazu und auf dieser Schicht bleibt das eigentliche Segment unsichtbar. Die Zwischenstationen (Netzwerkknoten) sehen nur die Informationen des Datagramms.

Die **Sicherungsschicht** ist für die Übermittlung von ganzen Rahmen im Netzwerk zuständig. Die Aufgabe kann man mit einem Postangestellten vergleichen, der den Brief an den entsprechenden LKW weiterleitet, der ihn zur Postverteilerstelle in der Zielstadt bringt.

Die **Bitübertragungsschicht** überträgt einzelne Bits/Signale aus dem Rahmen von Knoten zu Knoten.

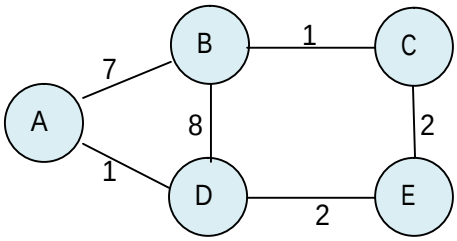
3. Das **Routing** betrifft die Vermittlungsschicht. Um die Pakete von einem Host (Knoten im Netzwerk) zu einem anderen Host zu schicken benötigt die Vermittlungsschicht eine Route, über die die Pakete geschickt werden sollen. Hier die beste Route zu finden, ist die Aufgabe des Routing-Protokolls. Diesem Routing Protkoll liegt ein Routingalgorithmus zu Grunde. Stellen wir uns folgendes Netzwerk vor, in dem es die Knoten A-F gibt. Dargestellt werden kann ein Netzwerk mit Hilfe eines Graphen. Die Knoten im Graphen stellen Router dar, also Punkte an denen Entscheidungen über den zu wählenden Weg getroffen werden. Die Linien zwischen den Knoten werden Kanten genannt, die Zahlen an den Kanten geben die Kosten für das Versenden eines Paketes an. Das Versenden eines Paketes von A nach D kostet also weniger als das Senden eines Paketes von D nach C.



Der kürzeste Pfad von A nach C wäre in diesem Beispiel der Weg über D also ADC mit Kosten von 4.

4. Grundsätzlich kann man Routing-Algorithmen in zwei Arten unterteilen. *Globale Routing-Algorithmen* berechnen den Pfad zwischen der Quelle und dem Ziel mit Hilfe der globalen Kenntnis über das Netzwerk. Bei einem *dezentralen Routing-Algorithmus* wird die Berechnung des Pfades mit den geringsten Kosten verteilt durchgeführt. D.h. kein Knoten verfügt über die vollständige Information des Netzes, sondern jeder Knoten kennt nur die Kosten zu den Nachbarknoten. Durch Austausch der Informationen mit den Nachbarknoten kann jeder Knoten die Nachbarknoten informieren. Da sich dieser dezentrale Algorithmus für eine Animation besser eignet als der zentrale Routing-Algorithmus werden wir diesen in der Animation anhand eines kleinen Beispiels durchsimulieren.

Zuerst soll aber ein Beispiel einer Distanztabelle für den Knoten E helfen, den Distanz-Vektor-Algorithmus zu verstehen:



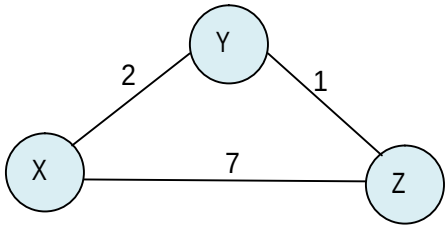
	$D^E()$	Kosten über A	B	D
Ziel	A	**1**	14	5
	B	7	8	**5**
	C	6	9	**4**
	D	4	11	**2**

```
In jedem Knoten, X:
   Initialisierung:
      Für alle Nachbarknoten v:
         D*(*,v) = ∞ // * bedeutet für alle Zeilen
         D^X(v,v) = c(X,v)
      Für alle Richtungen, y
         Schicke min_w D(y,w) zu jedem Nachbarn

   Solange (wahr)
         Warte (bis sich Kosten zum Nachbarn ändern oder der Nachbar ein
         Update schickt)

         Wenn (sich Kosten zu allen Zielen über Nachbar v um d ändern)
            Dann erhöhe die Kosten für alle Richtugnen um d

         Sonst Wenn (sich der kürzeste Pfad von V zu Y geändert hat,
         Update von V)
            Erhöhe die Kosten um den neuen Wert

         Wenn es für ein Ziel neue Kosten gibt
            Schicke den neuen Wert für die minimalen Kosten an die
            Nachbarn
```

Wir kennen also zu Beginn nur die Distanz zu unseren eigenen Nachbarn und erhalten dann in weiterer Folge von diesen die Informationen zu deren Nachbarknoten.

5. Animation: Für die Animation verwenden wir ein einfaches Netz aus drei Routern X, Y und Z. In AB-5.2 finden Sie für die Knotenpunkte die leeren Distanztabellen.

   Unser Netz sieht folgendermaßen aus:



Nach dem Initialisierungsschritt sehen die Distanztabellen für die 3 Knoten X, Y und Z folgendermaßen aus. Die Knoten kennen jeweils nur die Distanz zu ihren Nachbarknoten. Fett geschriebene Zahlen zeigen die aktuellen minimalen Kosten an.

Kosten über

$D^X()$	Y	Z
Ziel Y	**2**	∞
Ziel Z	∞	**7**

Kosten über

$D^Y()$	X	Z
Ziel X	**2**	∞
Ziel Z	∞	**1**

Kosten über

$D^Z()$	X	Y
Ziel X	**7**	∞
Ziel Y	∞	**1**

Im nächsten Schritt werden die Distanztabellen aktualisiert, nachdem die Nachbarknoten sich gegenseitig informiert haben. Die Minimalen Kosten von X zu Z haben sich somit von 7 zu 3 geändert, da es günstiger ist, über Y zu Z zu gelangen als direkt von X nach Z.

Kosten über

$D^X()$	Y	Z
Ziel Y	**2**	8
Ziel Z	**3**	7

Kosten über

$D^Y()$	X	Z
Ziel X	**2**	8
Ziel Z	9	**1**

Kosten über

$D^Z()$	X	Y
Ziel X	7	**3**
Ziel Y	9	**1**

Die Aktualisierungen gehen so lange weiter, bis es keine Aktualisierungsnachrichten mehr von den Nachbarn gibt. Dann geht der Algorithmus in einen Ruhezustand und die Knoten warten, bis sich die Kosten an einer Verbindungsleitung ändern. Dann werden natürlich wieder die Nachbarn informiert.

Die untenstehende leere Tabelle steht den Knotenpunkten zu Beginn der Animation zur Verfügung (siehe V-AB5.2).

Kosten über

$D^X()$	Y	Z
Ziel Y		
Ziel Z		

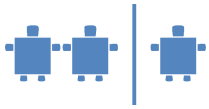
Kosten über

$D^Y()$	X	Z
Ziel X		
Ziel Z		

Kosten über

$D^Z()$	X	Y
Ziel X		
Ziel Y		

Jeder Router hat eine Tabelle für seinen Knotenpunkt und trägt somit ein, welche Kosten zum Nachbarknoten entstehen und informiert anschließend die Nachbarknoten darüber, welche Kosten zu den anderen Nachbarknoten entstehen. Wird z. B. durch den ÜL eine Änderung der



Kosten vorgenommen, muss die Tabelle geändert werden und die Nachbarknoten müssen über die Änderungen informiert werden.

Weiterführung: *Führen Sie eine Änderung der Kosten durch. Z.B. könnten Sie die Kosten zwischen X und Y auf 1 setzen und jene von X auf Z auf 50, um den Schülern auch den Aktualiersungsvorgang zu zeigen.*

Weiterführende Literatur

Kurose, James; Ross, Keith: *Computernetze. Ein Top-Down-Ansatz mit Schwerpunkt Internet*. Pearson Studium. Münschen, 2002.

Gallenbacher, Jens: *Abenteuer Informatik. IT zum Anfassen von Routenplaner bis Online-Banking.*

Spektrum Akademischer Verlag, München, 2007.