

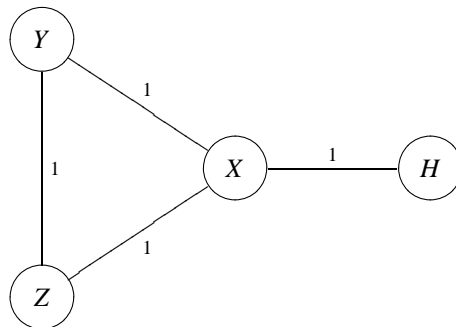
## 7. Aufgabenblatt zum Internet-Praktikum

### Aufgabe 1: (40 Punkte) Routing-Algorithmen

- (a) Verwende den Distance-Vector-Algorithmus **mit *poisoned reverse***, wie er im Buch von Kurose und Ross: *Computer Networking* im Abschnitt 4.2.2 beschrieben ist, um die Routing-Tabellen von X, Y und Z in folgender Topologie schrittweise von Hand<sup>1</sup> zu berechnen. Dabei sind jeweils nur die Einträge mit Ziel H aufzuschreiben. Nimm an, dass alle Router synchron arbeiten. In jeder Zeiteinheit führt jeder Router folgende Schritte aus:

- Senden von Updates an seine Nachbarn (Übertragungsdauer: 0 Zeiteinheiten)
- Empfangen der Updates von den Nachbarn (die folglich in der gleichen Zeiteinheit abgeschickt wurden)
- Updaten der eigenen Routing-Tabellen.  
(Der Inhalt der upgedateten Tabellen bestimmt dann den Inhalt der Updates, die in der nächsten Zeiteinheit verschickt werden.)

Die initialen Routing-Tabellen für X, Y und Z sind nachfolgend angegeben. Gib alle weiteren Routing-Tabellen an, bis ein stabiler Zustand erreicht ist.



$D^X$	H	Y	Z
H	$\infty$	$\infty$	$\infty$
$D^Y$	X	Z	
H	$\infty$	$\infty$	
$D^Z$	X	Y	
H	$\infty$	$\infty$	

$D^X$	H	Y	Z
H			
$D^Y$	X	Z	
H			
$D^Z$	X	Y	
H			

$D^X$	H	Y	Z
H			
$D^Y$	X	Z	
H			
$D^Z$	X	Y	
H			

$D^X$	H	Y	Z
H			
$D^Y$	X	Z	
H			
$D^Z$	X	Y	
H			

$D^X$	H	Y	Z
H			
$D^Y$	X	Z	
H			
$D^Z$	X	Y	
H			

$D^X$	H	Y	Z
H			
$D^Y$	X	Z	
H			
$D^Z$	X	Y	
H			

$D^X$	H	Y	Z
H			
$D^Y$	X	Z	
H			
$D^Z$	X	Y	
H			

<sup>1</sup>Naja, Du darfst auch ein Programm dafür schreiben, wenn Dir das lieber ist ...

- (b) *Poisoned reverse* eliminiert nicht in allen Fällen das *count-to-infinity*-Problem. Gib ein Beispiel an, bei dem *poisoned reverse* versagt. Verwende hierzu obige Topologie und wähle einen geeigneten Link, der bei einem Ausfall unter Anwendung von *poisoned reverse* zu *count-to-infinity* führt. Schreibe das Ergebnis aus (a) in die folgende Tabelle, und nimm im nächsten Schritt den Ausfall des Links an. Zeige den Fortschritt der Routing-Tabellen während vier Zeiteinheiten (also insgesamt  $5 \times 3$  Tabellen inkl. der Start-Tabellen).

$D^X$	H	Y	Z	$D^X$	H	Y	Z	$D^X$	H	Y	Z	$D^X$	H	Y	Z
H				H				H				H			
$D^Y$	X	Z		$D^Y$	X	Z		$D^Y$	X	Z		$D^Y$	X	Z	
H				H				H				H			
$D^Z$	X	Y		$D^Z$	X	Y		$D^Z$	X	Y		$D^Z$	X	Y	
H				H				H				H			

$D^X$	H	Y	Z	$D^X$	H	Y	Z	$D^X$	H	Y	Z	$D^X$	H	Y	Z
H				H				H				H			
$D^Y$	X	Z		$D^Y$	X	Z		$D^Y$	X	Z		$D^Y$	X	Z	
H				H				H				H			
$D^Z$	X	Y		$D^Z$	X	Y		$D^Z$	X	Y		$D^Z$	X	Y	
H				H				H				H			

**Aufgabe 2:** (60 Punkte) *Implementation einer einfachen Forwarding-Table*

(60 Punkte) Da Flooding eine sehr ineffiziente Methode zum Weiterleiten von Nachrichten ist, soll in dieser Aufgabe ein einfacher Mechanismus zur Optimierung der Nachrichtenweiterleitung implementiert werden. Dieser Mechanismus besteht aus einer einfachen Tabelle, deren Einträge mit Hilfe empfangener Nachrichten generiert werden und es erlauben, Nachrichten an bekannte Ziele direkt in die korrekte „Richtung“ weiterzuleiten. Diese Tabelle wird auch *Forwarding-Table* oder *Forwarding Information Base (FIB)* genannt.

Die Tabelle soll folgendermaßen aufgebaut und benutzt werden:

- Das Protokoll muss so erweitert werden, dass Textnachrichten mit einer TTL (Time To Live) versehen werden:  
100 MESSAGE ID <client-ID>:<message-ID> CHANNEL <channel-ID> LINES <Anzahl Zeilen>  
TTL <time-to-live>  
Die TTL wird benutzt, um Weglängen abschätzen zu können. Der Absender einer Nachricht initialisiert die TTL mit 10. Jedesmal, wenn eine Text-Nachricht weitergeleitet wird, wird der TTL-Wert um 1 verringert. Wenn der TTL-Wert auf 0 sinkt, muss die Nachricht verworfen werden.
- Die Client-ID muss nun folgendermaßen aufgebaut sein:  
<IP-Nummer> „:“ <Portnummer> oder  
<Rechner-Name> „:“ <Portnummer>  
z. B. viper:24769 oder 131.159.14.137:24697 oder xyz.bla.laber.suelz.de:24967
- Server sollen für ihre ID die Portnummer des Listen-Sockets (!) verwenden.
- Aus jeder empfangenen Text-Nachricht sind die TTL und die Client-ID des Absenders zu extrahieren.
- Mit Hilfe dieser beiden Werte sollen Einträge in die Forwarding-Tabelle gemacht werden, die folgende Informationen enthalten:
  - Timestamp: wann wurde dieser Eintrag zuletzt aktualisiert
  - Ziel: zu initialisieren mit dem Absender
  - Direction: zu belegen mit der Verbindung (Socket), von der die Nachricht gelesen wurde
  - Metric: zu belegen mit dem TTL-Wert der Nachricht

Diese Tabelle enthält also für jeden Knoten, von dem man eine Nachricht empfangen hat, die Verbindung, über die man selbst für den Sender der Nachricht erreichbar ist und somit auch, wie man den Sender selbst erreichen kann<sup>2</sup>.

<sup>2</sup>Abgesehen von der TTL entspricht dieses Verfahren übrigens dem, das in lernenden Ethernet-Switches genutzt wird.

- Tabellen-Einträge sind so zu pflegen, dass immer der Weg gespeichert ist, über den man den höchsten TTL-Wert empfangen hat (und der deshalb der kürzeste ist). Wege mit niedrigeren TTLs sind zu ignorieren, Wege mit gleicher TTL sind dagegen einzutragen (weil aktueller).
- Wenn in der Forwarding-Table kein Eintrag für ein gegebenes Ziel zu finden ist, dann soll die Nachricht wie vorher geflooded werden.
- Die Tabellen-Logik soll so gestaltet werden, dass das Pflegen (Einfügen, Updaten, Löschen) über Funktionsaufrufe möglich ist, da der Mechanismus zum Erlernen von Routen im nächsten Blatt verfeinert werden wird.

Abzugeben ist der kommentierte Quellcode des erweiterten Server-Knotens.

---

**Details zur Abgabe: siehe FAQ (unterhalb <http://www.net.in.tum.de/teaching/SS05/inetprak/>).**

**Abgabedatum: Dienstag, 7. Juni 2005, 23:59h s.t.**