Zusammenfassung Computernetzwerke und verteilte Systeme

TECHNISCHE UNIVERSITÄT DARMSTADT

Alex Praus, Tobi Kratz
5. Februar 2021

Inhaltsverzeichnis

1 Quick Tour

1.1 Making two devices communicate

Verschiedene Möglichkeiten, um Kommunitkation herzustellen: direkte physische Verbindung. Hier werden Daten als Bits übertragen. Eine 1 könnte als als steigende Taktflanke und eine 0 als fallende dargestellt werden, es gibt jedoch viele Probleme bei der Implementation (00, 11, wechselnde Taktfrequenz...).

low level properties of communication Es gibt verschiedene Werte, die bei low-level communication wichtig sind:

- Delay (Latenz) d = distance / Propagation speed v. (v ist im vakuum = c, in Kupfer etwa $\frac{2}{3}$ c)
- Data rate r (Datenrate) = Data size/ data rate (Bsp. bits/secound).
  Wichtig ist hier, dass nicht die Geschwindigkeit gemeint ist, mit der die Daten Transportiert werden (siehe v), sondern in welcher rate die Bits auf die Leitung gelegt werden.

Wenn der Sender eine Daten sendet, werden diese nicht auf Sender Seite gespeichert, sonder lediglich beim Empfänger. Allerhöchstens sind die Daten während der Übertragung im Kabel gespeichert.
Beispeil Latexberechnung (delay & data rate): Wir senden 1250 Bytes ($10^4 b$) über 6 Meter mit einer Geschwindigkeit von 10 Mbps ($10^7 \frac{b}{s}$). Der Delay d beträgt dabei $d = \frac{6m}{c} = \frac{6m}{3 \cdot 1 - 8 \frac{m}{s}} = 2 \cdot 1 - ^{-8} s = 20ns$. Die Data rate beträgt $r = \frac{10^4 b}{10^7 \frac{b}{s}} = 10^{-3} s = 1ms$.

Types of physical communication Die Typen lassen sich in 3 Fälle aufteilen:

- Simplex: Eine Seite kann nur senden, die andere nur Empfangen (one-way). Empfänger lassen sich jedoch beliebig skalieren.
- Half Duplex: Beide Seiten wechseln sich ab mit senden um Empfangen (vgl. Telefonat/Gesrpäch).
- Duplex: Beide Seiten können senden wie und wann sie lustig sind (vgl. Streitgesrpäch).

Während Simplex und Half Duplex einfach realisierbar sind, ist (Full) Duplex eine technische Herausforderung.

Realizing Half Duplex Denkbar wären 2 Kabel, eins je Host, das wäre jedoch eine Verschwendung, da diese Kabel nie Zeitgleich genutzt werden würden. Es gibt zwei Ansätze: Time divison duplex (TDD) und on-dmand duplex. Beim TDD hat jeder Host eine feste Zeit T zum senden, und es wird zwangsläufig abgewechselt. Beim on-demand duplex gibt jeder Host die Länge der nächsten Bit Sequenz am Anfang direkt bekannt (pre-announce).

Realizing Full Duplex Hier wären 2 Kabel eher anwendbar, bedeuten aber den doppelten Aufwand. Auf kurzer Distanz kann jedoch auch Full Duplex mit einem Kabel realisiert werden, in dem Jeder Host eine leicht andere Frequenz benutzt (z.B. bei WiFi). Auch TDD ist umsetzbar. Während A sendet, speicher B die zu sendenden Daten. Nach T sendet B dann den Stack während A speichert. Das ist jedoch nicht Analog umsetzbar, da z.B. Sprachdaten sich nicht in Portionen aufteilen lassen.

1.2 Connecting many computers

Jeden Computer direkt mit jedem anderen zu verbinden wäre zwar möglich, skaliert jedoch eher so semi (https://www.reddit.com/r/cablegore/top
Switches wäre eine Lösung, jedoch laufen dann mehrere Connections über ein Kabel, es kommt also zu Einbußung in der Data Rate. Außerdem besteht das Problem, wie man eine Connection über einen switch herstellt (vgl. Vermittlung beim Telefon). Das führt jedoch dazu, dass eine Connection nur einfach genutzt werden kann, ist ein Host also mit einem anderen verbunden, kann ein Host nicht mehr erreicht werden.

Packet switching Statt also ein Circut für eine Verbindung zu blocken, teilt der switch die Daten in packete und sendet diese. So wird die Leitung nur für die Länge eines Packets geblockt und es kann schneller gewechselt werden.
Probleme: Anfang und Ende bestimmen? Wie bestimmt man wo das Packet hin soll?
Beispiel Ablauf: »store-and-forward«switching:

1. recieve a complete packet
2. store the packet in a Buffer
3. Find out the packet's destination
4. decide where the packet should be sent next (benötigt Kenntnis über Netzwerk Topologie)
5. forward the packet to his next hop of its journey

Multiplexing

> »Organziging the forwarding of packets over such a single, shared connection is called multiplexing.«

Auch beim Multiplexing ist TDM (Time Division Multiplexing) (nur ein Packet gleichzeitig) und FDM (Frequenz Division Multiplexing) (mehrere Packete auf gleichzeitig auf unterschiedlichen Frequenzen) möglich. Bei optischen Verbindungen WDM (wavelength division multiplexing) statt FDM. Es gibt jedoch noch weitere Formen, die hauptsächlich für Wireless transmission geeignet sind: CDM (Code Divison Multiplexing) und SDM (Space Division Multiplexing).
Multiplexing lässt sich auch abstrahieren, wenn zum Beispiel mehrere connections eines höhrer Layers eine lower-level connection nutzen sollen (s.c. upward multiplexing). Allgemein lässt sich in dem Kontext von shared Recources sprechen.

Forwarding and next hop selection Bekanntes Problem: Wie weiß eine Host/router/switch, wo er Packete hinschicken soll? Wie kennt er die besste/schnellste Verbindung? Es gibt einfache Ansätze:

- Flooding: alle Packete an alle Nachbarn senden
- Hot-potato routing: So schnell wie möglich an einen/mehrere zufällige Nachbarn senden

Sinnvoller wäre jedoch, wenn sich der router die besten wege merkt, e.g. durch Routing Tabellen. Diese können durch 2 Arten gesammelt werden: aktiv und passiv.
Passiv: Aktiv traffic beobachten und daraus Schlüsse ziehen.
Aktiv: Informationen aktiv senden und empfangen unter den verschiedenen Routern (routing protocols).
Jedoch bei Netzen der größe unsere Internets können Routing Tabellen schnell sehr groß werden. Hier kann man das Netz in mehrere Teilnetze splitten (devide et impera).

1.3 Organizing the mess - and connecting 'Alien' computers

Der Schlüssel ist »Simplification by abstraction«.
Z.B. das Modell des DS (distributed Systems) hilft beim verstehen von CN (computer networks). ein DS besteht aus AS (autonomous systems) und CSS (communication subsystems). Oft referiert wird auch auf die Abstraktion des OSI (Open Systems Interconnection) Schichten Modell.

1.3.1 OSI

Part 1: Concepts and Terms

> »A protocol defines the format and the order of messages exhanged between two or more communicating entities, as well as the actions taken on transmission and/or reception of a message or other event«

Analog zum 'Computer' Protokoll kann man sich ein menschliches Protokoll vorstellen, z.B. der definierte Ablauf beim Telefonieren (Hallo, Hallo..... Tschüß, Tschau). Weiter ist das OSI Modell in schichten aufgeteilt. Analog wäre z.B. Layer 2 ein Manager, der einen Brief in Auftrag gibt (Layer 1), der dann von der Post (Layer 0) transportiert wird. Ein layer N hat bietet also services für den layer n+1 an. Zwei layer n kommunizieren nur mit den für diesen layer vorgesehenden Protokolle, diese können jedoch auf services von layer n-1 zugreifen (der unterste layer kann natürlich nur eigene services nutzen).
Services im OSI Modell lassen sich in 2 Gruppen aufteilen:

- **connection-orientated services:** Diese haben meist 3 Phasen:
  - CON (Connection Establishment)
  - DAT (Data Exchange)
  - DIS (Disconnect)
- **connectionless services:** Bei diesen fallen CON und DIS weg und es werden nur Daten Ausgetauscht.

Im Osi Modell werden Nachrichten höhrer Layer als Daten Einheiten (Data Units) tierfer Layer transportiert. Hierfür gibt es einigen common Notations:

- packet: Ist die Einheit, die Transportiert wird (kann aus Fragmenten bestehen)
- datagram: wird bei connectionsless services als ersatz für Packete verwendet
- frame: 'fertig und verpackt' um versendet zu werden.
- cell: kleinere packet mit einer definierten größe
- PDU (Protocol Data Unit): eine (N)-PDU ist definiert durch: (N)-PCI+(N)-SDU
- PCI (Protocol Control Information): wird nur von peers genuzt.
- SDU (Service Data Unit): Ist die zu versendete payload eines höheren layers. (N)SDU=(N+1)-PCI+(N+1)-SDU

Part 2: 7 Layer Model

Layer 1: physical Layer (PH) Senden von Bits durch (de-)aktivieren von Signalen auf Leitungen.
Beispiel: 1000 Base-T

Layer 2: data link layer (D) Sendet Packete als Frames um Fehler zu erkennen und zu beheben, um Fehleranfällige Hosts zu schützen. Kann auch Flow Controll verwenden, um langsame Hosts zu schützen.
Beispiel: Ethernet

Layer 3: network layer (N) Ziel ist es den Packet-Stream zwischen zwei Hosts zu ermöglichen. Koordinierung der Pfade von Host zu Host. Konkret: Routing Wege finden und Packete weiterleiten und Fehler beheben, z.B. durch 'flow controll'. Up- und downward multiplexing ist möglich. Außerdem kann congestion control auf diesem layer angewendet werden.
Beispiel: IP

Layer 4: transport layer (T) Logische Verbindungen zwischen zwei Prozessen (nicht nur zwischen zwei Computern), Fehlerkorrektur und Paketzusammensetzung für den N-Layer.
Beispiel: TCP

Layer 5: session layer (S) Koordiniert Session z.B. bei HTTP mit dem Session-Cookies und hilft Nutzer und Anwendung bei der Konstruktion und dem spannen von aufeinanderfolgenden Verbindungen.
Beispiel: HTTPS

Layer 6: presentation layer (P) Sellt die Daten in eine unabhängige Form um, um unahängig davon die Übermittlung - gg. mit Kompression und Verschlüsslung - zu ermöglichen.
Beispiel: LDAP

Layer 7: application layer (A) Stellt Funktionen wie Daten Ein- und Ausgabe zur Verfügung.
Beispiel: XMPP

1.3.2 5 Layer of the Internet

Im Internet verschmelzen Layer 5,6,7 oft und oft sind die übergänge nicht klar definiert.

Layer 1 übermittlung von frames als stream von bits

Layer 2 Daten von layer 3 in Frames verpacken und and direkte Nachbarn weiterleiten

Layer 3 Daten vom client zum web-server weiterleiten, router2router communication, außerdem können e2e Verbindungen durch hop-to-hop realisiert werden

Layer 4 Verlässliche Verbindung zum Web-server herstellen und sicherstellen, dass die Daten auch in der richtigen Reihenfolge ankommen, jedoch keine congestion control.

Layer 5,6,7 HTTP Anfragen erstellen, Ebene 4 aufrufen (TCP).

1.3.3 network types

Wie schon erwähnt gibt es CO und CL networks. Beispiel für CO ist z.B. Das Telefon, CL die Post. CO haben durch die durch Handshakes vorallem bei kurzen Verbindungen eine Hohe 'extra Last' durch die zusätzlichen Daten des Handshakes, lassen sich jedoch besser skalieren, da sie nicht einen Status gebunden sind (stateless). CO sind jedoch durch den Status der connection Zuverlässiger. Bei einem 'verstopften' Network können mit CL immernoch Daten versendet werden, es kann nur sein, dass diese verspätet ankommen, CO haben jedoch schwierigkeiten. Es ist möglich CO auf CL aufzubauen.

connection-oriented Networks Im CO network ist der erste Schritt mit einem handshake eine connection herzustellen. Nach dem handshake wissen beide Seiten von der connection und der Datenaustausch kann stattfinden. Die connection ist dabei nur ein loser status, auf dem Basis jedoch andere Eigentschaften (flow control, congestion control...) aufgebaut werden können.
Bei CO networks implementiert nicht direkt andere Eigentschaften der Verbindung. Dinge wie reliability, flow control und congestion control sind für CO networks sind nicht notwendig. Diese können z.B. mit TCP ermöglicht werden.

connectionless networks keine handshakes, direkt Faken (Daten Austausch). Wenn der Datenaustausch vorbei ist, kommt auch kein DIS mehr. CL networks brauchen wenig Aufwand, da keine connection gepflegt werden muss, es kann jedoch sein, dass der reciever nicht bereit zum Empfangen ist. CL implementieren keine reliability, flow control und congestion control.

2 Routing

2.1 introduction

Im Routing werden dafür gesort, dass Packete vom Empfänger an das richtige Ziehl kommen. Bei Direkt Verbindungen besteht das Problem nicht, jedoch ist das bei großen Netzwerk keine Option, wenn jeder Host mit jedem anderem Verbunden werden muss. Wenn Switches genutzt werden, muss diesen jedoch gesagt werden, wie das Netzwerk aufgebaut ist (Netzwerk Topologie). In diesem Kapitel geht es um das Problem, wie ein Host den besten Weg zu einem Ziehl findet.

Building a large network Bei großen Netzwerken ist flooding und Hot-potato routing keine Option, da mit jedem Host die Anzahl an Packeten steigt und so mit den beiden Routing uneffizienten Methoden das Netzwerk schnell an sein Limit kommt. Ziel ist es eine Effiziente Methode zu finden, die Packete möglichst schnell ans Ziel bringt, ohne dabei unnötig viel Traffci erzeugt.
Im folgenden werden zwei Begriffe genutzt:

- **Routing:** determine route taken by packets from source to destination. (Basis: Routing algorithms).
- **Forwarding:** move packets from router's input to appropriate router output.

2.1.1 Forwarding

Wenn Packete von einem Netzwerk in ein anderes Netzwerk geileitet werden sollen, wird ein router eingesetzt. (Heute haben uns bekannte Router mehrere Aufgaben, die früher von verschiedenen Geräten übernommen wurden: hub, bridge, switch, gateway.) Wenn also Packete von einem Netzwerk in ein anderes gesendet werden sollen, übernimmt der Router die Koordination und leitet das Packet (forwarded) in das entsprechende Ziel Netzwerk. Hängt das andere Netzwerk direkt am selben Router, handelt sich sich um ein single hop. Wenn mindestens 2 Router zwischen den Netzwerken sind, handelt es sich um ein Multi-Hop.

2.1.2 Routing

Routing findet für gewöhnlich auf Layer 3 statt, dort ist das Ziel Packete von Host A zu Host B möglichst effizient zu transportieren bzw. erstmal einen Pfad zu finden, auf dem das möglich ist. Dies wird i.d.R. von Routing Algorithmen durchgeführt. Das Internet besteht aus mehreren AS, die alle wieder aus Teilnetzen bestehen. Jedes AS führt dabei selbst routing Algorithmen aus, um die besten Wege zu finden.

- CONS (CO+NS) Nn CO Networks Routing Algorithmen werden meist in der CON Phase durchgeführt. Im COTS (CO+TS) wissen nur die Endsysteme, dass sie verbunden sind, in CONS hingegen wissen alle Systeme auf der Route, dass die Systeme verbunden sind.
- CLNS (CL+NS) IN CL Networks wird nicht bei jedem Packet der Algorithmus durchgeführt, das würde einen zu großen Overload bedeuten. Manchmal wird beim ersten Packet einer Verbidnung der Algorithmus durchgeführt, das ist allerdings für sich schnell änderte Netzwerke keine Option. Im Inetnet z.B. dies in regelmäßigen Abständen, oder wenn sich große Teile ändern.

optimizing Routing Algorithmen Routing Algorithmen haben oft unterschiedliche Kriterien, nach denen sie arbeiten:

- Average packet delay
- Total throughput
- individual delay (kann jedoch mit anderen Kriterien im Widerspruch stehen)

Am meisten jedoch wird nach dem Kriterium, des minimal-hop-count gearbeitet, da dieser oft einen Kompromiss aus allen Kriterien bedeutet, es gibt jedoch keine Garantie dafür.

2.2 Routing Algorithms

Routing Algorithmen werden meist in zwei Arten aufgeteilt:

- **Non-adaptive Routing Algorithms**
  Diese agieren unabhängig vom State des Netzwerks. Beispiele sind flooding oder preconfiguration.

- **Adaptive Routing Algorithms**
  Nehmen den aktuellen Status des Netzwerks mit in Betracht, wenn sie Routing Entscheidung treffen. Beispiel Hierfür wären distance-vector-routing oder link state routing. Das Problem hierbei ist, dass bei sich ändernden Neztwerken die Routen häufig neu entschieden werden. Algorithmen diesen Types sind trotzdem sinnvoll in eignen, fest bekannten Netzen. Bekommt ein Link z.B. so viel Traffic, das Packete verloren gehen, wird der Link als Broken makiert und es kommt zu noch größeren Ausfällen.
  Es gibt dort auch 3 Unterarten:
  - Centralized adaptive routing
  - Isolated (aka. local) adaptive routing
  - Distributed adaptive routing

2.2.1 Examples

Flooding (non-adaptive) Hier wird jedes einkommende Packet an alle bekannten Nachbarn weiterleitet. Das Problem dabei ist, dass so Nezte schnell überlastet werden. Gibt es zum Beispiel schleifen, kann es schnell zu einer Flut an nicht aufhörenden Packeten kommen. Eine Lösung für dieses Problem wäre z.B. das implementieren von TTL (Time to live) oder Sequence Number. TTL werden meist in Hops angegeben und werden bei jedem Hop um 1 dekrementiert. Hat ein Packet ein TTL von 0, wird es weggeworfen. Eine Sequence Number wird beim ersten Router initialisiert. Jeder Router führt eine Tabelle mit Sequence Numbers, die er schon einmal geroutet hat. Kommt ein Packet mit einer Sequence Number, die er schon kennt, wird dieses Packet weggeworfen.
Flooding macht jedoch durchaus Sinn in sich schnell ändernden Netzen, z.B. bei WLAN oder Mobilfunk oder wenn alle Packete Multicast sind und so wie so mehrere Ziele haben.

Static Routes (non-adaptive)Static Routes sind großartig für statische, vorhersehbare Umgebungen. Das Problem ist, dass sich das Internet regelmäßig ändert und statische routen dann viel Wartungsaufwand bedeuten.

Centralized Adaptive Routing (adaptive) Es gibt einen Zentralen Control Center (RCC), der regelmäßig Informationen über die Topologie von allen Routern bekommt und dann einen Idealen Routing Graph erzeugt (z.B. Dijkstra). Das Problem hierbei ist, dass das Netz zusammenbricht, wenn nur der RCC ausfällt. Außerdem werden Routen 'in der Nähe' des RCC bevorzugt, was dort zu einer hohen Last führt während 'abgelegene' Router meist wenig Aufgaben haben. Ebenso bekommen Router die näher am RCC sind schneller die neuen Routing Informationen, was zu unterschiedlichen States führen kann.

Isolated (aka. local) adaptive Routing (non-adaptive) Es werden Entscheidungen über Routen nur lokal getroffen. Beispiele sind Hot potato Routing und Backward learning.

- Hot Potato Routing
  Idee ist es, die Packete so schnell wie möglich loszuwerden, wobei nicht beachtet werden muss, zu welchen Host die Packete geschickt werden. Dieser Algorithmus ist nicht sehr effektiv, es gibt jedoch einige use-Cases in denen diese Art noch genutzt wird (peering/discovering).

- Backward Learning Routing
  Bei diesem Algorithmus werden im Packet Header Source Addresse und Hop Counter hinzugefügt, Router lernen also im laufenden Betrief über die Topologie und passen die Routen im Betrieb an. Jedoch müssen in jungen Netzwerken andere Algorithmen genuztzt werden (z.B. hot potato / flooding). Wenn der Hop-Count == 1 ist, kommt das Packet von einem direkten Nachbarn. Bei einem Hop-Count n>1 ist die source n hops away.

Distributed Adaptive Routing Durch Graph Abstraction die besten Routen finden. Knoten sind dabei Router und Kanten die physikalischen Links a.k.a. hops. Die Kosten eines links sind dabei z.B. delay, $, oder der congestion Layer. Die Kosten eines Pfades sind dann alle link Kosten vereint. Ein guter Pfad wird meist als der, mit den geringsten Kosten bezeichnet, es kann aber auch nach anderen Kriterien gesucht werden (e.g. min-hop-count).
Algorithmen hier lassen sich weiter klassifizieren:

- **Decentralized** Jeder Router kennt die Kosten zu seinen Nachbarn. Auch Distance Vector Routing gehört hierzu (z.B. BGP oder RIP)
- **Global** Alle Router kennen die komplette Topologie und alle link kosten. Hierzu gehören Link state Algorithmen z.B. Dijkstras oder OSPF.
- **Static** (nicht adaptiv) Routen ändern sich sehr selten
- **Dynamic** (adaptiv) Routen können sich oft ändern, Hier werden also regelmäßig updates in den Routen gemacht.

2.3 Distance Vecotr Routing

Beim Distance Vector Routing tauschen direkte Nachbarn Informationen über Routen mit ihren Nachbarn aus. Jeder Host pflegt eine Tabelle, in der jede mögliche Ziehladdresse eine Reihe und jeder Nachbar eine Spalte hat. In der Tabelle werden dann die "Kosten"der Route eingetragen und mit jeder Iteration verbessert. Konkret schreibt man dann für Route von X to Y via Z als nächsten Hop:

$$D^X(Y,Z) = c(X,Z) + min_w\{D^Z(Y,w)\}$$

Mit einem Routing Algorithmus wird dann eine "Distance Table/Matrix"gebaut, mit der dann Routing Tablelen aufgestellt werden, aus denen dann der Distance Vector an die Nachbarn announced werden kann. DVR hat jedoch einige Probleme (count to infinity), jedoch handelt es sich um einen sehr simplen Algorithms.
DVR Protokolle sind iterativ und Distributed:

- **Iterativ** Das heißt sie laufen nicht unendlich, sondern stoppen sobald keine weiteren Verbesserungen möglich sind. Außerdem sind sie ßelf-terminating"d.h. es gibt kein Stop Signal o.ä. Eine Iteration wird dabei ausgelöst indem entweder ein loakler link sich ändern z.B. in den Kosten oder wenn es eine Nachricht eines Nachbarn gibt, dass der Link dort sich geändert hat.
- **Distributed** Des weiteren tauschen sie Informationen nur mit direkten Nachbarn aus und kennen auch nur den State dieser. Eine Node informiert einen Nachbarn dann über neue Routen, wenn sich die Kosten zu einer Destination verringert haben.

2.3.1 Count to Infinity Problem

Gegebene Situation: Wir haben 3 Host A,B,C. A ist mit B mit einem Cost von 1 verbunden, und B ist mit C mit einem Cost von 2 verbunden. Daraus ergeben sich folgende 3 Routing Tabellen: Durch einen Ausfall verschwindet jetzt die Verbindung zwischen B und

A			B			C		
TO	COST	VIA	TO	COST	VIA	TO	COST	VIA
B	1	B	A	1	B	A	3	B
C	3	B	C	2	C	B	2	B

C. A announced an B jedoch, dass es eine Route zu C mit dem Cost von 3 gibt. B versucht nun also C via A zu erreichen: Nachdem B

A			B			C		
TO	COST	VIA	TO	COST	VIA	TO	COST	VIA
B	1	B	A	1	B	A	-	-
C	3	B	C	4	A	B	-	-

dann diese Information an A sendet, aktualisiert A dann seine Route zu C, da diese über B geht und sich die Kosten erhöht haben. die Route von A nach C via B ist dann wie gewohnt die Route von B nach C + die Kosten von A nach B. A announced das dann wieder an B, der ja nach C über A routet. Er addiert darauf also die Kosten von B nach A. Das läuft dann ungebremst so weiter, bis ins unendliche,
Möglichkeiten dieses Problem zu lösen:

Poisend Reverse Methode Wenn die Route von A nach C über B geht, sagt A dem Host B, dass seine Kosten nach C unendlich sind. In einem kleinen Netzwerk wird dann innerhalb weniger Iterationen ein stabiler State erreicht, in größeren Netzwerken besteht das Problem jedoch immernoch, z.B. in einem Netzwerk in dem A,B,C jeweils direkt verbunden sind, und C dann noch eine Verbindung zu D hat. Alle Kosten sind gleich. Fällt dann die Verbindung CD aus, bekommt A immernoch Falsche Routen zu D von B und umgekehrt.

Split Horizon Wenn Host B seine Routen updatet und das an A sendet und A daraufhin einige Änderungen übernimmt, sendet A diese Änderungen nicht wieder an B, sondern nur an seine anderen Nachbarn.

2.4 Link-State Routing

Beim Link State Routing sammelt für gewöhnlich ein Zentraler Knoten (RCC) Informationen und gibt diese dann an alle anderen Router im Netzwerk weiter. Die Netzwerktopologie ist somit dann allen Router im Netzwerk bekannt. Der RCC baut aus allen gesammelten Informationen einen Grapgen (V,E), wobei V ein set an vertices (nodes) ist und E für die Edges (links) steht. c(v,w) sind dann die Kosten der Kanten. Wenn eine Kante nicht in E ist, ist c unendlich. Das Ziel ist es dann den günstigsten Pfad von node s (source) zu node v zu finden. Hierfür wird meistens Dijkstras verwendet. Jeder Router versendet regelmäßig per flooding ”Link state packages” mit Informationen, die er zu seinen Nachbarn gesammelt hat (delay, hop count...) und verseht diese mit einer sequence Number und einem ”age flag”. Wenn ein Router so ein Packet bekommt, das er jedoch schon kennt (sequence Number) oder es abgelaufen ist, wirft er es weg.

Vergelich LSR und DVR Link State Routing und Distance Vecotr Routing im direkten Vergleich:

	LSR	DVR
Message complexity	mit n Knoten und E Kanten werden jedes mal $O(n \cdot E)$ nachrichten versendet	Austausch findet nur zwischen nur zwischen Nachbarn statt
Speed of Covergence	ein $O(n^2)$ Alogorithmus braucht $O(n \cdot E)$ Nachrichten	Variiert stark. Es kann zu Routing Schleifen kommen. Count-to-infinity Problem
Robustness	Es können verfälschte Link Kosten announced werden. Jede Router nutzt nur die eigenen Tabellen.	Es können verfälschte Path Kosten announced werden. Die eigenen Routen werden von anderen Routern genutzt.

Algorithmen wie LSR und DVR sind für Netze konzipiert, die sich selten ändern und physikalisch verbunden sind. Sie haben vorallem Schwächen bei Mobilen Netzen, bei z.B. folgenden Punkten:

- **High dynamics** z.B. durch stendig wechselnde Nachbarn und Links
- **Power conservation** regelmäßiges senden von Routing Packeten verbraucht Strom und Leistung
- **Low bandwidth links** wenn z.B. Routing Informationen nicht versendet werden können
- **Asymmetry** Links können in der Geschwindigkeit variieren
- **Interference** Störsignale
- **High redundancy** ein Gerät ist mit vielen anderen Verbunden / "meshed"

2.5 Hierarchical Routing

In der Realität sind große Netze nicht so ideal wie bisher beschrieben, sondern sind meist nicht flach wie wir sie beschrieben haben und Router unterscheiden sich oft fundamental. Im Internet heute gibt es über 1 Milliarde links, die alle zu erreichen würde Routing Tabellen explodieren lassen und der Austausch dieser Tabellen würde jeden Link sprengen. Deswegen ist das Internet in Teilnetze aufgeteilt: Autonomous Systems.

Autonomous Systems Jedes AS hat eine eigene Nummer (z.B. AS421220) und kennt eine Route zu jedem anderen AS. Es gibt im heutigen Internet etwa 60000 solcher Teilnetze und alle sind unterschiedlich groß. Verschiedene AS sind mit pysikalischen Links verbunden (peers), über die Daten ausgetauscht werden. Jeder Router innerhalb eines AS muss also nur die Routen zu anderen Routern im AS kennen, und die Route zu seinem Gateway. Innerhalb eines AS (inra-AS) hat der Administrator freie Wahl für die Nutzung von Routing Algorithmen, es kann z.B. RIP, OSPF oder IGRP verwendet werden. Für die Kommunikation ziwschen AS (Inter-AS) gibt es jedoch einen festen Standart: BGP (Border Gateway Protocoll).
Für die inter-AS Kommunikation wird ein Gateway Router benötigt, der den Transfer von Daten zu anderen AS koordiniert. Dieser Router pfelgt Routing Tabellen mit anderen Gateayrouter. Der Vorteil dieses Modells ist, dass es für das Reale Internet besser skaliert. Durch die reduzierung von Peers, die in Routing Tabellen gepflegt werden müssen kommt es zu selteren updates der Tabellen, was es leichter macht, schnelle Routen zu finden. Bei Inter-AS Routing gibt es jedoch Regulierungen, welcher Host über wen peeren darf, innerhalb eines AS wird es sowas nicht geben, da ein AS nur eine begrenzte Anzahl Admins hat. Inter-AS Communication kann deswegen eher Performance orientiert sein.

2.5.1 About BGB and Internet Routing

Zum Vegleich die Routing Protokolle RIP & OSPF:

- **RIP**
  - DVR
  - Erstmal 1983 aufgetaucht. RFC 1058 von 1988
  - Am minimalen Hop-Count orientiert
  - Poison Reverse
- **OSPF**
  - LSR
  - RFC 1131 (inzwischen Version 2 & 3)
  - am meisten verwendete IGP
  - verwendet TCP zum Übertragen von Routing Informationen
  - Multicast support

Border Gateway Protocoll BGP ist der heutige Standart für EGP. Durch regelmäßige "hello"Packete erfahren andere Netzteilnehmer von der existenz des AS. BGP peers bauen dann eine Session auf und tauschen via TCP Routing Informationen aus. Wenn AS1 z.B. AS2 sagt, dass es einen gewissen prefix routen kann, garantiert AS1 AS2 dann allen Traffic weiterzuleiten. BGP kann auch Innerhalb eines AS verwendet werden, man spricht dann von iBGP. Um den Unterschied dann zu inter-AS Communication zu spezifizieren, wird in dem Kontext dann von eBGP gesprochen.
Eine advertiste Route in BGP besteht immer aus einem Prefix und einem Attribut set. Die zwei wichtigsten Attribute sind

- AS-PATH: Beinhaltet den Pfad, der für die Route genutzt wird (z.B. AS5 AS8 AS10)
- NEXT-HOP: Der nächste AS Router für den nächsten HOP

Über die Jahre wurden immer mehr AS registriert und die Zahl steigt weiter (>6000). Das führt dazu, dass es wieder mehr Host gibt, für die alle Routing Einträge gepflegt werden müssen, was die Länge der Tabellen explodieren lässt. Durch die Hohe AS Zahl, kommt es auch regelmäßiger zu Änderungen im System, was Änderungen der Routing Tabellen beudeutet. Große Netze haben zudem eine höhere Fehleranfälligkeit.

BGP Security In BGP stellt jeder AS selbst ein, welches Subnetz er verwaltet. Kommt es bei dem einstellen zu Fehlern ein AS announced ein Subnetz, welches er nicht wirklich organisiert, kommt es zu falschen Routen. So geschehen z.B. 24.02.2008 der Pakistan Telekom (AS17557). Eine generelle Einschätzung zu Routing Security siehe RFC4593. Mit BGP können aber auch Security Operations implementiert werden, z.B. durch setzen einer IP TTL von 255 und es werden nur Routing Informationen mit einer TTL >= 254 verarbeitet. Außerdem können BGP Session mit MD5 Signaturen versehen werden.

2.6 Mobile Routing

Mobile Networking ist gut geeignet, um ein schnelles Netzwerk an Orten ohne gut Infrastruktur aufzubauen. Anders als im Internet sind beim Mobile Routen ganz andere Probleme zu bewältigen. Da zwischen Host keine physikalische Verbidnung besteht ist, wird Routing zwischen Host schwierig. Doch durch sich wechselnde Positionen o.ä. verändern sich links zwischen Host sehr schnell. Das und andere sind Faktoren, mit denen DVR und LSR nicht umgehen können, diese sind für statische Netze konzipiert. Routing Algorithmen können in zwei Kategorien aufgeteilt werden:

- **Proaktiv**
  - Routing Informationen werden unabhängig vom aktuellen Traffic regelmäßig und undabhängig generiert
  - Alle Ïnternet Routing Algorithmenßind Proaktiv, auch die oben kennengelernten Beispiele
  -
  - Beispiel für ein Reactive Mobile Routing Algorithmus: DSDV
- **Reactive**
  - Routen werden erst dann berechnet, wenn Daten übertragen werden sollen
  - Bei CO wird eine Route (wenn noch keine Vorhanden ist) beim connection Setup berechnet
  - Bei CL beim ersten Packet
  - Beispiel für ein Reactive Mobile Routing Algorithmus: DSR

Hierarchical Mobile Routing In Mobilen Netzen können Ideen des Mobile Routings verwendet werden. Teilnetze können in Cluster aufgeteilt werden. Innerhalb eines Clusters kann proaktives Routing implementiert werden, für die Kommunikation zwischen Clustern können reaktive Algorithmen verwendet werden. Bei kleinen Clustern können alle Nodes sich gegenseitig kennen, es ist also ideal für reaktive Routing. Zwischen Clustern findet seltener Kommunikation statt, und es können mit vielen kleinen Clustern hier am besten reaktive Algorithmen verwendet werden.

2.6.1 DSDV

DSDV ist eine Erweiterung des DVR spezialisiert für Mobile Netze. Es gibt 2 große Extensions:

1. Sequence Numbers for all Routing updates
   - Verbesserungen gegen Schleifen und Inkonsistenzen
2. Decrease update frequency
   - Zeit zwischen ersten und bestem announcemendd eines Pfades wird gespeichert

Trotz dessen ist DSDV noch ein Proaktiver Algorithmus und ähnelt sehr stark dem DVR.

2.6.2 DSR

DSR (RFC 4728) geht einen Schritt weiter Richtung Reaktivem Routing. Er baut auf zwei simplen Ideen auf:

1. Routing wird ind "Path discoveryünd "path maintenanceäufgeteilt
2. regelmäßiges Updaten wird vermieden

Er ist für statische und dynamische Netze geeignet und kann mit bis zu 200 Knoten arbeiten. SSource Routing"bedeutet dabei, dass der Sender für die Bestimmung des Pfades verantwortlich ist.

Path Discovery Wenn ein Sender ein Packet versenden will, jedoch noch keine Route für das Ziel hat, startet er die Path Discovery. Es wird dabei ein Packet per flooding und boradcast mit der Ziel Addresse und einer einmaligen ID versendet. Wenn ein Host ein solches Packet erhält und er ist das Ziel, sendet er das Packet mit dem Pfad über das es gekommen ist zurück. Das erste Packet dass diesen Host erreicht kam über dem schnellsten Pfad, jedes weitere Packet kann dann verworfen werden. Es sind noch weitere Optimierungen möglich: Wenn die Topologie (bzw. die maximale länge) des Netzes bekannt ist, kann statt einer ID eine conter/TTL hinzugefügt werden. Der Host kann dann am Counter sehen, wie viele Hops das Packet hinter sich hat und es wegwerfen, wenn der Counter größer als der maximale Diameter ist. Eine zweite Verbesserung wäre, wenn Hosts discovery Packete speichern, die sie weiterleiten sollen. Die Informationen dieser Packete können dann für die Suche einer route genutzt werden, wenn das Gerät selbst Packete versenden will. Ist ein Path gefunden, muss jedoch sichergestellt werden, dass dieser auch über die Länge der Verbindung offen bleibt.

Path Maintenance Nicht genutzte Paths werden nach einer Zeit aus der Routing Tabelle gelöscht. Es ist möglichkeit, ist z.B. als Sender auf eine Bestätigung des Empfängers auf layer 2 zu warten oder so eine explizit zu erfragen. Auch kann eine Station schauen, ob Nachbarn das Packet weiterleiten, wenn diese Technologie unterstützt ist. Falls es ein Problem gibt, können Pfade neu gesucht werden oder es kann versucht werden, den Empfänger zu erreichen und ihm mitzuteilen, dass es ein Problem gab.

2.7 Overlay Routing

Im Overlay Routing sitzt ein Virtuelles Neztwerk auf einem existierenden. Hier können die gleichen Algorithmen verwendet werden. Wird ein Packet in einem Overlay network an den Nachbarn versendet, wird das Packet über das darunter liegende Netz transportiert und kann dort auch über mehrere Hops geleitet werden, während Sender um Empfänger im Overlay Network denken, sie sind dierekte Nachbarn.

3 Referenzen

3.1 Quicktour

- TDD (Time divison duplex)
- TDM (Time Division Multiplexing)
- FDM (Frequenz Division Multiplexing)
- WDM (wavelength divisionmultiplexing)
- CDM (Code Divison Multiplexing)
- SDM (Code Divison Multiplexing)
- DS (Distributed Systems)
- AS (autonomous System
- CSS (communication subsystem)
- CN (computer networks)
- OSI (Open Systems Interconnection)
- CON (Connection Establishment)
- DAT (Data Exchange)

- DIS (Disconnect)
- PDU (Protocol Data Unit)
- PCI (Protocol Control Information)
- SDU (Service Data Unit)
- LAN (Local area network)
- LLC (Logical Link Control)
- MAC (Medium Access Control)
- CO Network (connection-orientated network)
- CL Network (connectionless network)
- e2e (End-to-end)

3.2 Routing

- NS (network layer services
- TS (transport service)
- TTL (Time to live)
- RCC (Routing Control Center)
- DVR (Distance Vector Routing)
- LSR (Link-State Routing)
- AS (autonomous System)
- RIP (Routing Information Protocol)
- OSPF (Open Shortest Path First)
- IGRP (Interior Gateway Routing Protocol)
- BGP (Border Gateway Protocol)
- IGP (Interior Gateway Protocol
- EGP (Exterior Gateway Protocol)
- DSDV (Destionation Sequenced Distance Vector)
- DSR (Dynamic Source Routing)