Zusammenfassung Computernetzwerke und verteilte Systeme

TECHNISCHE UNIVERSITÄT DARMSTADT

Alex Praus, Tobi Kratz
2. 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:

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

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)