Telematik

2 Internet Congestion Control

2.1 Basics

Packet Switching

[Header | Header | User Data]

forwarded independently

General problem: n users on 1 resource for data transferring

Critical Situations: 2 packets arrive @ same time.

Buffer / drop?

Different speeds, data incoming faster than being able to send

First buffer packets, then drop

* fills up, Latency

End-to-end-Latency: propagation delay + transmission delay + queuing delay

TCP: 3-way handshake for connection establishment

4-way for termination (half-duplex possible)

Full duplex Connection

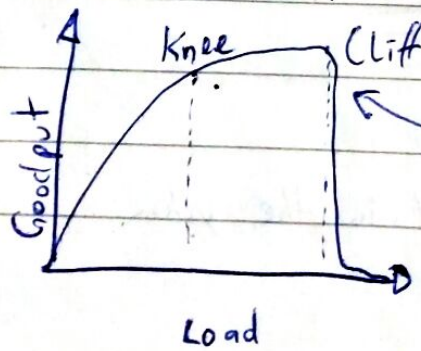
Byte oriented seq numbers

Go back N

positive cumulative acks

Timeouts

Flow control via sliding window



Network is congested, Buffers full

Goodput = Throughput - Overhead data

Flow control only applies to the capacity of the receiver
⇒ congestion control necessary due to possibility of multiple TCP connections on the same resource
Goal: rates @ knee
Feedback from the routers necessary
Congestion avoidance and congestion control: avoid traffic @ cliff

Efficiency: $C \hat{=}$ Capacity of resources

$$\sum_{\text{users}} \text{data rate} = C$$

Fairness: $F(r_j, \dots r_n) = \frac{(\sum r_j)^2}{N(\sum r_j^2)}$ Jain Fairness Index
between 0 and 1
$1 \hat{=}$ fair, equal data rate $1/N$ unfair

Convergence, responsiveness, rate at which speed a connection gets to equilibrium

I Smoothness, oscillations around equilibrium state at steady rate

Distributedness
No central control with knowledge about network state

Window based Congestion control
Cwnd $\hat{=}$ Congestion control window
unacknowledged Packets allowed in the system.

Rate based
UDP, controled by precise timers

Others: hop-by-hop, end-to-end, router based, centralized

Implicit due to timeouts, duplicated acks
Explicit, indicated by the network nodes, ECN

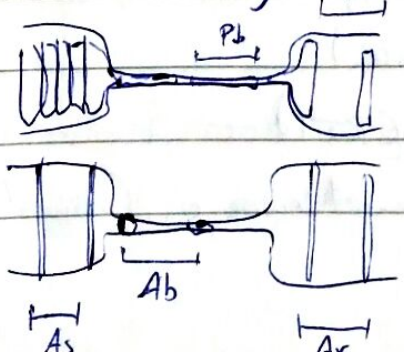
TCP Tahoe
Mechanismen: Slow start (bei Timeout)
Fast retransmit (bei dup. ACKs)
Timeout
Congestion avoidance

Es muss immer gelten
LastByteSent − LastByteACKed ≤ min{Cwnd, RcvWindow}
RcvWindow ≙ Empfangsfenster (Flusskontrollfenster)
SSThres, wann wird von Slow Start zu Congestion avoidance übergegangen?
Additive Increase, Multiplicative Decrease

Timeout oder 3 dup ACKs: Stau vermutet
SSThres = max{FlightSize/2, 2*MSS}
# unacknowledged Packets

Fast retransmit bei der 3. Duplizierung (4. ACK)

Self-Clocking



P: Zwischenankunftszeit zweier Pakete
A: Ankunftszeit zweier ACKs
b: Bottleneck
r: receiver
s: sender

Sender sendet Pakete so schnell er kann, der Empfänger sendet ACK so schnell sie ankommen (bei Bottlenecks langsamer), wonach der Sender nur in diesem Intervall neue Packete senden darf → Bottleneck bestimmt Geschwindigkeit der Übertragung

3 ways to fail Conservation of packages:
1. Connection does not get to equilibrium
2. Sender injects new Package before old has left (⇒ ¬Conservation!)
3. Resource along path hinder equilibrium

Slow Start
bei vermuteter Stausituation oder neuer Verbindung erhöhe Cwnd um 1 für jedes erhaltene ACK, bis SSThresh erreicht ist → exp. Wachstum

Maximum segment size (MSS) bezieht sich auf die Nutzdaten (in Bytes)

Go-Back-N: Sendewiederholung ab erstem unbestätigten Segment, Bursts von wiederholten Segmenten

Retransmission Timer
Alternative zum erniedrigen der Flightsize (außer per ACK, Paket hat System verlassen)
Basiert auf der Round-Trip-Time (RTT)
Estimated RTT basiert auf mehreren Messungen (Samples)

Estimated RTT = $(1-\alpha) *$ Estimated RTT $+ \alpha *$ Sample RTT

α typischerweise 0,125

Zeitgeber für Sendewiederholung

Retransmission Timer = $\beta *$ Estimated RTT

β oft = 2

Schwankung an der RTT durch sich ändernde Last im System

Deviation = $(1-\gamma) *$ Deviation $+ \gamma * |$ Sample RTT − Estimated RTT $|$

Retransmission Timer = Estimated RTT $+ \beta *$ Deviation

γ oft = 0,25

Mehrfache Sendewiederholungen

Wie groß ist der Zeitraum zwischen zwei Sendewiederholungen des gleichen Segments?

Exponentieller Backoff:

Retransmission Timer *= 2, max 60 sec.

Karn's Algorithmus bei Quittierung eines wiederholten Segments:

Estimated RTT & Deviation nicht neu berechnen.

Backoff weiterhin verdoppeln, bis Quittung eines nicht wiederholten Segments eintrifft. Dann alten Algo wieder nutzen.

Unnötige Sendewiederholungen vermeiden => Timer müssen akkurat sein.

Packet loss highly likely due to new TCP connection.

Reduce load of "our" active connection.

Feedback control Algorithm AIMD

applied to congestion control

$$r_j(t+1) = \begin{cases} r_j(t) + a & \text{if no congestion is detected} \\ r_j(t) \times b & \text{if congestion is detected} \end{cases}$$

$a \in \mathbb{N}$, $b \in [0,1) \subset \mathbb{R}$

Assuming all users receive the same feedback from the bottleneck link => equal resource utilization
∧ MSS equal

Fairness of two connections:

$$F(r_j, r_i) = \frac{(r_i + r_j)^2}{2(r_i^2 + r_j^2)}$$

Optimal operating point: $(C/2, C/2)$

Gieriger Nutzer öffnet mehrere Verbindungen gleichzeitig oder sendet mehr ACKs, bzw. schneller als die Daten wirklich eintreffen

Dies wäre nicht möglich, wenn das Staukontrollfenster in Bytes zählen würde. Es zählt in Segmenten

Quittungen fälschen wäre unrentabel, wenn Sender Pakete auslässt. Die dann gelogenen „angekommenen" Daten würden dann fehlen.

Appropiate Byte counting

Zähler für quittierte Datenmenge

bytes_acked += # Bytes in Quittung quittiert

if bytes_acked ≥ Cwnd [Bytes]:

Cwnd += 1 MSS

bytes_acked -= Cwnd [Bytes]

Dies hilft gegen Senden mehrerer Quittungen pro Segment

TCP Reno = TCP Tahoe + Fast ~~Retransmit~~ Recovery

bei Leichter Stausituation (Dup. ACKs) kein Slow Start

Fast Recovery

Sendwiederholung des Segments (Fast Retransmit)

SSThresh = max{FlightSize/2, 2*MSS}

Cwnd = SSThresh + 3 MSS

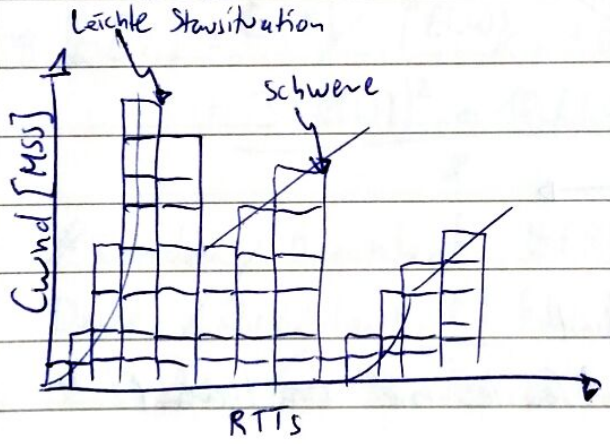
Bei Empfang duplizierte Quittung

Cwnd += 1

Sende (falls vorhanden) weitere noch nicht gesendete (neue) Segmente

Bei Empfang neuer Quittung:

Cwnd = SSThresh



Periodic Model

Zur Analyse von TCP (beispielsweise)

ist TCP für meine Zwecke geeignet?

Welche Leistung / welches Verhalten kann ich erwarten?

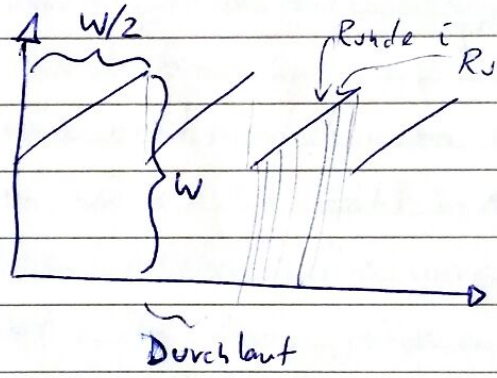
Betrachtung eines langfristigen Verhaltens (SS wird ignoriert)

RTT Konstant. Verluste treten periodisch auf.

Kein Queuing delay. Kein Warten auf timeouts

⇒ Staukontrollfenster bleibt die perfekte periodische Sägezahnkurve.

Variablen: X: Übertragungsrate, Segmente/Zeit
W: Größe des Staukontrollfensters bei Paketverlust
N: # Segmente
A: Dauer eines Durchlaufs
RTT: Umlaufzeit
p: $\mathbb{P}$(Verlust eines Paketes)
MSS: Maximum Segment Size
$\mathbb{E}$: Erwartungswert
O: Größe der zu übertragenen Daten



W_{cur} = aktuelle Größe des Cwnd
Annahme: Flusskontrollfenster nie limitierend

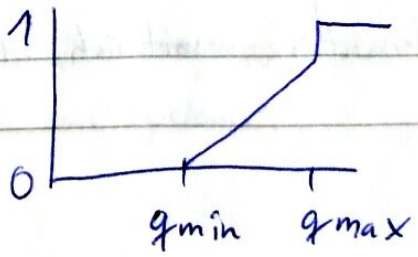
Größe minimales Staukontrollfenster: $W/2$
Wachstum um je 1
Länge der Durchläufe: $W/2 * RTT$
Segmente pro Durchlauf: $3/8\ W^2$
Es gilt $N = 1/p$
$\Rightarrow W = \sqrt{8/3p}$
Durchschnittliche Übertragungsrate: $1{,}22 * \frac{1}{RTT\sqrt{p}}$
Bsp. $RTT = 200ms$, $p = 0{,}05$ $\Rightarrow X = 27{,}4$ Segmente/s

Detailiertes Paketverlustmodell
Verluste nicht mehr periodisch
Durchläufe nicht gleich lang
$\alpha_j \hat{=}$ Nummer des verlorenen Segments in Runde R_j des i-ten Durchlaufs
In R_j + Runden: $Y_j = \alpha_j + W_j - 1$ Segmente gesendet
$\Rightarrow \mathbb{E}[Y] = \mathbb{E}[\alpha] + \mathbb{E}[W] - 1$
Erwartete Position des verlorenen Segments:
$\mathbb{E}[\alpha] = \sum_{k=1}^{\infty} k(1-p)^{k-1} p = 1/p$
$\Rightarrow \mathbb{E}[Y] = \frac{1-p}{p} + \mathbb{E}[W]$
$\mathbb{E}[W] = 2 * \mathbb{E}[R] - 2$
Segmente im Umlauf: $Y_i = \frac{R_i}{2}\left(\frac{W_{i-1}}{2} + W_i\right) + \beta_i$
$\beta_i =$ # Segmente im Umlauf in der letzten Runde
$\Rightarrow \mathbb{E}[Y] = \frac{\mathbb{E}[R]}{2}\left(\frac{\mathbb{E}[W]}{2} + \mathbb{E}[W]\right) + \mathbb{E}[\beta]$
$= \frac{3(\mathbb{E}[W])^2 + 10\,\mathbb{E}[W]}{8}$

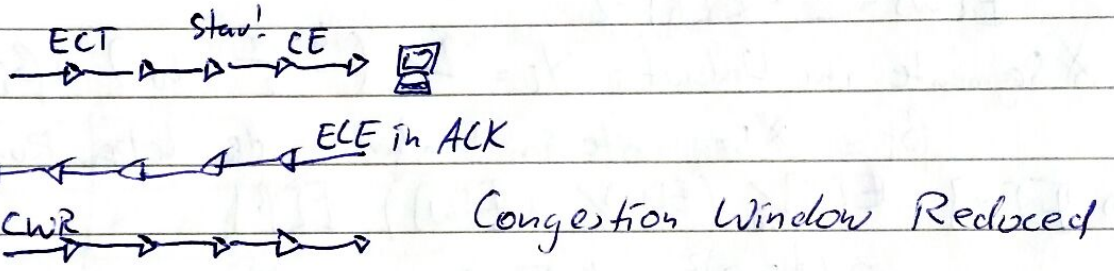
Runden / Durchlauf: $\mathbb{E}[R] = 5/6 + \sqrt{2/3 p - 23/36}$
Dauer / Durchlauf: $\mathbb{E}[A] = RTT * \left(11/6 + \sqrt{2/3 p - 23/36}\right)$
$\Rightarrow X(p) = 1/RTT * \sqrt{3/2p}$

Im Beispiel: $RTT = 200\,ms$, $p = 0{,}05$ $\Rightarrow X(p) = 23{,}89$ Segmente/s

Active Queue Management (AQM)
RED (Random early detection) $\hat{=}$ verwerfe zufällig ein Packet, falls sich die Buffer zunehmend füllen.
Drop probability:



ECN (Explicit Congestion Notification)
Leite Paket normal weiter, allerdings kann eine ECN gesetzt werden, sollten die Buffer sich zunehmend füllen - Diese ECN lässt den Sender wissen, dass er sein Cwnd zu reduzieren hat.
Dafür werden 2 bits im ToS field genutzt
00 Not-ECT, 01 ECT(1) 10 ECT(0) 11 CE
CE ≙ Congestion Experienced ECT ≙ ECN (capable Transport
Fähigkeit über ECN wird beim TCP Verbindungsaufbau geklärt.



3. Router

Auf Schicht 3, Vermittlungsschicht.
Dienen zur Wegewahl im Kommunikationssystem umgesetzt mit Hilfe der Weiterleitungstabelle im Router
leitet Pakete in Richtung Ziel weiter
tauscht Informationen mit anderen Routern aus, nutzt dazu Routingprotokolle
⇒ verteiltes & adaptives Routing
Erstellt Routing- und Weiterleitungstabelleneinträge
Je Paket ein Lookup
Pakete führen Informationen mit sich, IP Adresse, ...

Basisfunktionen:

Überprüfen des IP-Headers
- Versionsnummer
- Länge gültig
- CRC / Prüfsumme
- Lifetime check TimeToLive --;

Prüfsumme neu berechnen und setzen

Lookup, Suche nach dem korrekten Ausgangsport

Fragmentierung

Behandlung von IP-Optionen

Klassifizierung / Priorisierung

Line Speed

Routers need to keep up for every packet size

@ 100Gbit/s dauert ein ACK 3,2ns (40 bytes)

$$\frac{320\,bit}{10^{11}\,bit/s} = 3{,}2ns$$

~50% of all packets transmitted are <100 bytes long

Core Router

Used by service providers

need to handle large amount of data / traffic

fast lookup & forwarding

redundancy for reliability, cost secondary issue

Enterprise Router

connects end systems in companies, universities, ...

large number of end systems

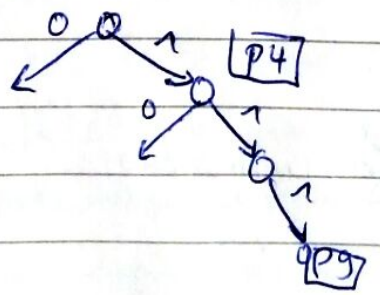
Support for firewalls, VLAN
Low cost per port, large number of ports

Edge router / access router
At edge of service provider
provide connectivity to customer from home, small business
Support for PPTP, IPsec, VPNs

Forwarding
done by prefix matching
longest select the most specific match
can be done in software
efficient data structures required
to perform a fast lookup
should be memory efficient. Core routers have
> 500k entries
Fast updates >> 100/sec since route changes occur frequently

N = # prefixes
W = length of a prefix
k = length of a stride (only for multibit tries)

Binary Trie

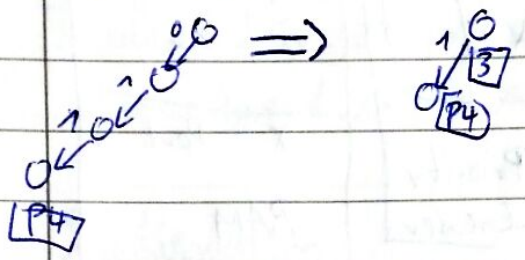


Lookup O(W)
Memory O(W·N)
Update O(W)

Path compression

Long sequences of one-child nodes waste memory

store the position of the bit that next decides which branch should be taken.



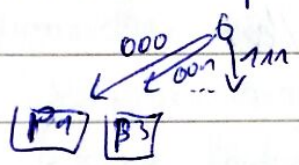
lookup $O(W)$

memory $O(N)$

updates $O(W)$

Multibit Trie

match multiple bits at a time instead of a single bit



Stride = 3 $\Rightarrow 2^3$ branches

$n \Rightarrow 2^n$

lookup $O(W/k)$

memory $O(2^k NW/k)$

update $O(W/k + 2^k)$

Hash Tables

Can perform lookup in $O(1)$

Used to store the trie lookup result

Does an entry for a destination exist in the hash table?

yes → no trie lookup ($O(1)$)

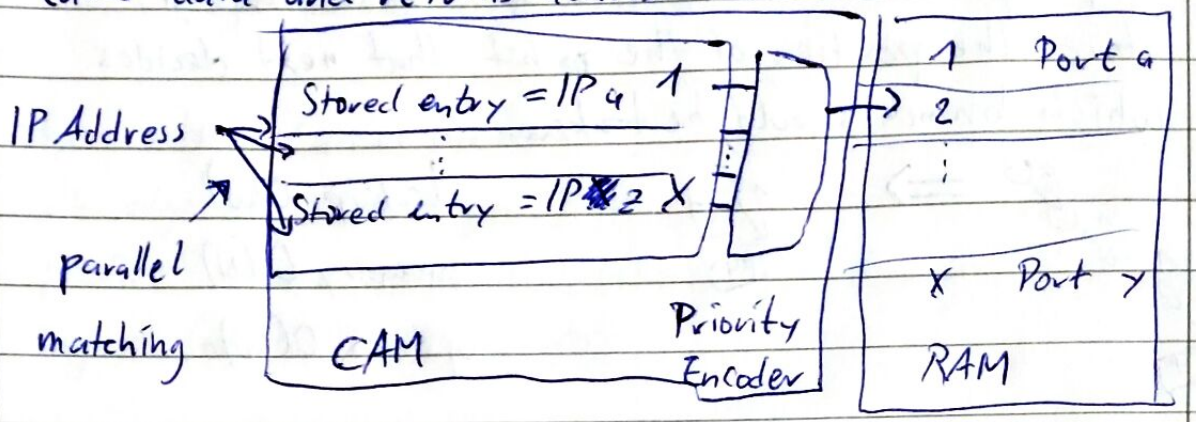
no → trie lookup

Longest Prefix matching in Hardware

IP address as RAM address → doesn't scale well, unused RAM

Content Addressable Memory (CAM)

takes data and returns address



Ternary CAM (TCAM)

Supports Don't Care state of bit

1001 1XX Ordering matters

1011 XXX

0XXXXXX

High energy consumption

2-3 times transistors like SRAM

strict ordering, updating values requires re-ordering

Router-Architektur

Netzwerkschnittstellen

realisieren den Zugang zum Netz

Schicht 1 & 2

Routing-Prozessor

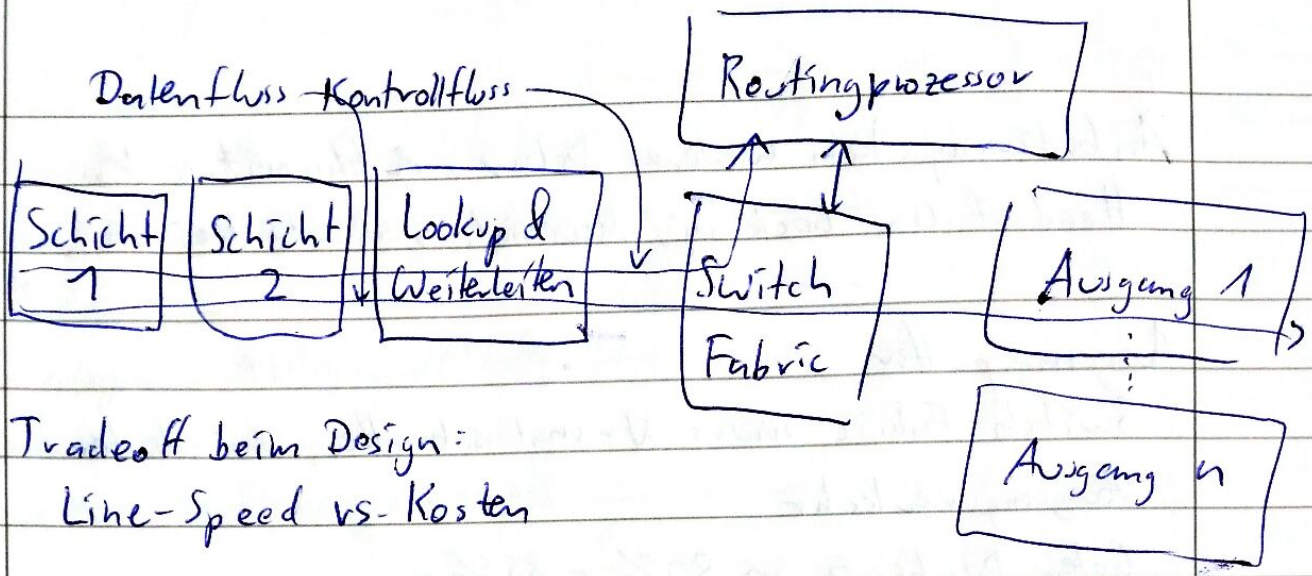
Routing Protokoll

Management Funktionalitäten

Switch Fabric

Backplane

Realisiert die interne Weiterleitung



Tradeoff beim Design:
Line-Speed vs. Kosten

Blockierungen
n Pakete an n Eingängen, die alle den selben Ausgang nehmen wollen.
Vermeidbar durch
Overprovisioning
Interne Verbindungen in Switch-Fabric optimieren mit höherer Geschwindigkeit als die Eingänge
Puffering
Pakete zwischenspeichern, bis die Resource frei ist.

Backpressure
Signalisierung der Überlastung am Eingangsport
Daraufhin die Last reduzieren

Parallele switch-fabrics
ermöglicht parallelen Transport z Ausgangsports
benötigt höhere Geschwindigkeit als an den Ausgängen

Platzierung von Puffern
Eingangspuffer
FIFO oder andere Schedules, Round Robin, Prioritäten

Anforderung: kein weiteres Delay, Zyklus zeit = 1/2
Head of Line bocking, Maximaldurchsatz bei 58%

Ausgangspuffer
Switch Fabric muss N-mal schneller sein als der Ausgangsverkehr
Guter Durchsatz ~ 80% - 85%

Verteilte Puffer
Switch Fabric als Matrix
Zyklus zeit = 1/2
Kein Head-of-Line-Blocking
Hoher Speicherbedarf

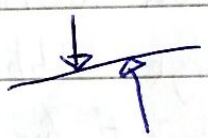
Zentraler Puffer
Alle Eingänge und Ausgänge mit einem Puffer verbinden.
Organisation: RAM
Zyklus zeit = 1/2N
Adress- und Steuerspeicher für Steuerung von parallelen Zugriffen
Niedrige Zugriffszeit nötig, Speicher dafür vergleichsweise klein

Switch Fabric
- Gemeinsamen Speicher
- Bus/Ring Struktur
- Koppelmatrix
- Mehrstufige Verbindungsnetze

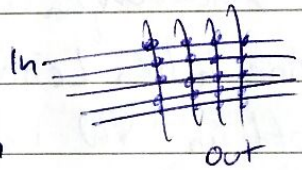
} typische Grundstrukturen

Bewertung der Grundstrukturen
Das interne Blockierungsverhalten
Puffer
Topologie, #Wege, #Stufen
Steuerungsprinzip
interne Verbindungskonzept

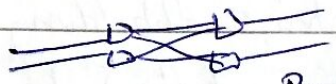
Bus / Ringstruktur
Konfliktfreien Zugriff durch Zeitduplex
Nötige Kapazität: mindestens Summe aller Eingänge
Einfacher Multi / Broadcast
Geringe #Ports ~ 16



Koppelmatrix
Vollvermaschung der Ein/Ausgänge
Teilweise parallele Vermittlung möglich
Puffer notwendig
Skaliert quadratisch
Eignet sich bei gleichgroßen Paketen



Mehrstufige Verbindungsnetze
Verdrahtungsaufwand geringer als bei Koppelmatrix
Jeder Eingang kann mit jedem Ausgang verbunden werden
Blockierungen möglich

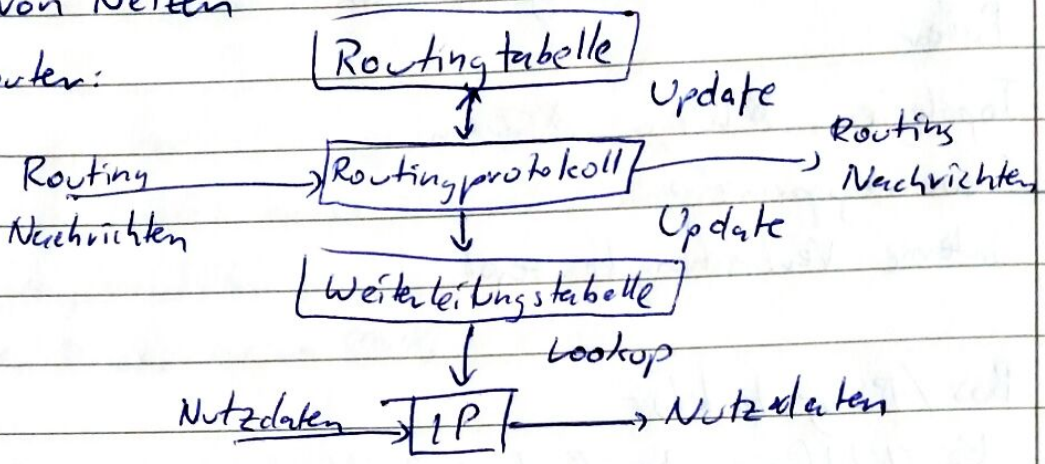


Banyan Netz

4 Internet routing

Netz von Netzen

IP Router:



Routingtabelle:

Von den Routingalgorithmen erzeugt

Einträge: Abbildung von IP Präfix auf next-hop (IP Adresse)

Optimiert für die Anforderungen der Routingalgorithmen

Änderungen der Topologie

Keine hohen Leistungsanforderungen

oft in SW realisiert

Weiterleitungstabelle:

Bei der Weiterleitung von Paketen genutzt

Einträge: Abbildung von IP-Präfix auf Interface (ID & MAC)

Optimiert für Suche nach Ziel-IP-Adresse

Hohe Effizienz erforderlich (Lookup in Line-Speed)

oft durch HW realisiert

Routing Metrik

Bewertungskriterium einzelner Übertragungsabschnitte

beeinflusst Bevorzugung / Vermeidung

Ganzzahl, unidirektional, Kosten / Gewicht

In Netzgraphen oft als Zahl an der Kante (Link), dann unidirektional gültig

Routing Policy
Betreibervorgaben zur Routing-Strategie
Üblicherweise im gesamten Netz gültig
Ermöglicht bevorzugen bestimmter Nachbarnetze bei der Wegewahl

Verteiltes adaptives Routing
Router tauschen Routinginformationen per Routingprotokoll aus.
Anpassung der Routen an die aktuelle Situation im Netz

Autonome Systeme
Protokolle innerhalb AS : Interior Gateway Protocoll (IGP)
" zwischen AS : Exterior " " (EGP)
Identifizierung durch 32 bit Nummer
Erscheint nach außen als eine Einheit
Besitzt einheitliche Routing Policy
" " IGP, die sich von AS unterscheiden können
Vorteile: Getrennte administrative Domänen
Skalierbarkeit durch 2 logische Ebenen
Routingprotokoll innerhalb eines AS und nicht weltweit
Zwischen AS Austausch von Routinginformationen

Overhead steigt mit der Größe des Netzes
Routinginformationen & auszutauschende Routing Informationen

erhöhen sich
Alle Router brauchen global gleiche Sicht
Distanz-Vektor-Algorithmen: Konvergenz problematisch

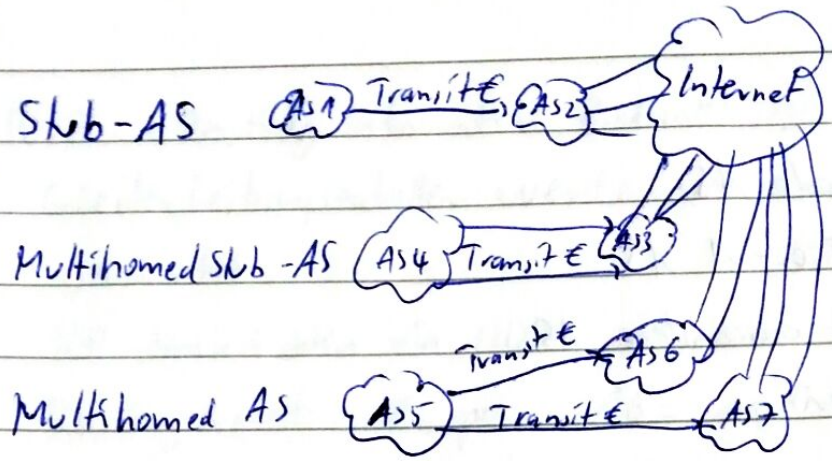
Betreiberautonomie
Auswahl des IGP
Verbergen der inneren Struktur

Stub AS
Meist kleine Unternehmen
Anschluss zu genau einem Provider
Kein Transitverkehr
} oft Kunde

Multihomed AS
Große Unternehmen
Anschluss an mehrere Provider
kein Transitverkehr
} oft Content-Delivery-Provider

Transit AS Tier-1
Provider
oft globale Ausdehnung

Konnektivität herstellen: baue das Netz aus. AS, das Internet
Transit: bezahlte Konnektivität
Upstream: Provider (Verkäufer von Transit)
Downstream: Kunde (Käufer " ")
Datenaustausch in beide Richtungen
bezahlt wird i. d. R. nur Downstream
Transit-AS
gehört einem Provider, der Transit bereitstellt



Peering ≙ direkte Verbindung zweier AS, aber kein Transit
weiter als 1 hop.
Vorteil: Beide AS sparen Transitkosten
Problem: skaliert Quadratisch

Public Peering
über IXP, zentrale öffentliche Stelle, an der Vernetzung stattfindet
neutrale Durchleitung auf Schicht 2
monatliche Kosten pro Port
(damit wird Wartung und Betrieb der Switching-Plattform gewährleistet)
Nutzung für Transit, Peering, VPLANs
Verschiedene Public Peering Policies:
open: AS ist offen für Peering mit jedem anderen AS
selective: nur mit Gewissen, unter bestimmten Bedingungen
restrictive: AS voll
no peering: AS betreibt kein Peering

Tier-1
verkaufen Transit
haben peering zu allen anderen Tier-1 AS
kein Einkauf von Transit

Tier-2
Große, überregionale AS
Downstream von Tier-1 AS
Verkaufen Transit an andere AS
Betreiber von Peering

Tier-3
Kleinere, Regionale AS
Downstream von Tier-2 AS
verkaufen keinen Transit an andere AS, nur an Anwender
betreiben Peering

Content delivery Provider (CDP)
Nähe zu Tier-1 Ports wünschenswert
Aufbau eines Content Delivery Networks (CDN)
Peering mit wichtigen Providern an zentralen Peering-Points

CDN
Weltweites Netz mit eigener ASN (AS Number)
Besteht aus Core Routern und Access Routern
Kunden verbinden sich mit " "
Lastbalancierung notwendig, " " sollen nahe am
Kunden sein

IGP
zur Weiterleitung innerhalb eines AS
meist: RIP oder OSPF, IS-IS oder EIGRP

RIP (Routing Information Protocoll)

Weiterleitungstabellen werden von einem Anwendungsprozess verwaltet

RIP-Nachrichten via UDP, periodisch

Routing Metrik: #hops, $16 = \infty$

alle 30 Sekunden

Timeout bei 180 Sekunden (setze Hop-Count auf 16)

Bei Änderung der Route, sende Triggered Update

Bei Empfang neuer Metrik < 15: Neuer Eintrag in Routing Tabelle

" " einer Nachricht, dessen Metrik kleiner ist, als ein aktuell gespeicherter Wert: aktualisiere Eintrag

In allen anderen Fällen: ignoriere Nachricht

OSPF (Open shortest Path first)

Link state Protokoll (Jeder Router muss den Zustand des gesamten Netzes kennen)

Dijkstra-Algorithmus zur Pfadbestimmung

Router versenden LSA (Link-State-Advertisement)

Inhalt: Besitzer, Links an Ports, Kosten = $\frac{\text{Reference Bandwidth}}{\text{Interface Bandwidth}}$

Jeder Router broadcastet seine LSAs und muss von allen anderen die LSAs empfangen können. Diese speichert er in einer LSD (Link State Database) und berechnet daraus seine Routen

Hello Protocoll

Contains ID of Router (sender) and expected receiver, if not known 0.0.0.0

If Router sees itself, communication is considered bi-directional

send periodically Hello Messages (10-30 sec)

LSA

header with information to uniquely identify LSA:
advertising router + Sequence number
body contains List of operational Links of the router:
Associated cost
Type: router / end-system
Router ID or IP Address
Are associated w/ Life time
incremented over time and invalid, if MaxAge is reached
often @ 1h
⇒ Routers must broadcast periodically ~ 30 min

Initial Synchronization

(re-)start of a router ⇒ empty LSD, may not participate in routing
immediately performs handshake with neighbouring router
hello protocol
link state information is then exchanged
considered adjecent

Coping w/ dynamic changes

if nothing changes, no messages are exchanged
only periodically after 30 minutes
what if Interface is down / router on the other site gets unreachable / Configuration changed?
minimum of time between LSA: 5 sec

Reliable Flooding
reception of LSA is acked by adjecent router
Multicast, hop limit = 1

LSA is considered "newer" and then stored, iff
(Seq is higher) OR ((Seq is Equal AND Checksum larger)
OR (smaller age))

If age is equal to maxAge and no LSA from sender is known
→ Send ACK & discard
If there is no LSA from sender or received LSA is newer
Store / replace LSA
Send ACK
Update age and flood LSA
If the stored copy is newer
Send stored copy to sender
If LSA and stored copy are the same
Treat it as ACK & discard

Scalability
AS can grow large
LSA flooding and computation overhead doesn't scale well
Idea: divide AS into areas, size ~100 routers
only flood and calculate routes for an area
Areas exchange summary-information w/ each other
⇒ hierarchy level in AS
Area 0 (backbone of AS)
Area Border Routers (ABR) interconnect areas

ABRs create summary-LSAs
it contains a list of reachable destinations within an area
each entry is associated with path cost from the ABR
Addresses are aggregated
after intra area path computation

Inter-Area forwarding goes through the backbone
⇒ end-to-end consists of source area → ABR
ABR → ABR
ABR → destination area
} always shortest paths

Broadcast links
full connectivity between routers → n^2 links
approach: associate LAN with a pseudonode which will receives broadcast
so called designated Router (DR)
" " Network LSAs

Types of LSAs
Router LSA: includes state of LSD
Network LSA: in multiaccess network, generated by the DR
Network-Summary LSA: generated by ABR, advertising directions outside areas
Opaque LSA: for Traffic engineering (TE) reasons

RIP	vs	OSPF
Metric: hop count		Addresses shortcomings of RIP
periodic updates, 30sec		Large Networks can be divided into areas
easy and memory saving, but problems with distance vector (count to ∞)		compatible w/ RIP

5 Label Switching

Issues with IP-based routing:
- lookup complex but needs to be fast
- TE difficult, always one shortest path used, not multiple shortest paths (if any)
- no support for data flows, each packet handled independently

Flow ≙ a set of packets that belong to the same interest
Microflow ≙ a single connection (TCP), many
Macroflow ≙ all TCP connections with destination of a given subset, fewer

Packet switching ≙ packets are forwarded independent of each other, metadata required, expensive forwarding
Circuit switching ≙ Connection with resource reservations, no metadata within the stream required (ISDN)
virtual circuits ≙ packets are forwarded by labels, connections do not have fixed resource reservation (MPLS)

Label switching is a combination of packet switching & circuit switchi
- packets are forwarded independently & need metadata (Label)
- paths established for flows through network
- simpler forwarding decision
- differentiation possible, TE, load balancing, QoS

Switching @ layer 2 instead of routing @ layer 3
Labels are only locally valid
Label ≙ short unstructured ID of fixed length
- unique within network, can be swapped
- circuit identified through sequence of labels on the path (→ virtual circuits)

Label can be placed on Layer 2,5 , 1 byte long, 20bit Label, TTL
or in the IPv6 header
Injected by an edge device (Label Edge Router (LER)) on the border of an label switching domain (LSD)
And removed before leaving LSD
Label swapping may occur within LSD, done by Label Switching Router (LSR)

Multi Path Label Switching (MPLS)
based on label switching
operates in data plane
Standardized within the LETF
increasingly adapted in large AS (faster than IPv6)
fast forwarding due to reduced packet processing
QoS support (for VoIP)
TE support to optimize network utilization
VPN support isolate traffic from other packets on the internet
Multiple Protocol support forwarding regardless of the underlying protocol (IP, ATM,!..)
Routing done by IP
Forwarding done by Label forwarding information base

Weiterleitungsklasse
Forwarding Equivalency Class (FEC)
Klasse von Paketen, die gleich behandelt werden sollen
~Synonym zu Flow
Gleicher Pfad (TE) ⇒ gleiche QoS
Grundlage für die Label zuordnung
Bsp. VoIP-Verkehr, gleiche IPs ~~und~~ mit gleichem Port

Granularität von Weiterleitungsklassen
Grob: wichtig für schnelle Weiterleitung und Skalierbarkeit
Fein: " " differenzierte Behandlung von Paketen
Vergleichbar mit einem Micro/Macroflow

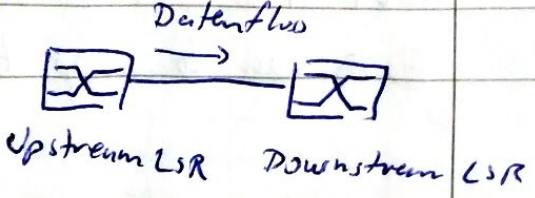
ToS-Typ + Quelladresse + Zieladresse + Quellport + Zielport ⇒ Label
→ Möglichkeit, Datenströme zu differenzieren

Weiterleitungstabelle besteht aus:
(=Label forwarding, ~~[illegible]~~ information base)
1. Eingehendem Label
2. Ausgehendem Label
3. Ausgehendem Interface
4. Adresse des nächsten hops

Was muss getan werden, damit MPLS läuft?
- definier FEC
- verteile Labels (von außen, traffic engineer)
vergleichbar mit dem Verteilen von Ratinginformationen

Label Bindings
stored in Label-Forwarding-Information-Base
used as incoming label
distributed to neighbouring routers

Roles:
Downstream LSR, in direction of dataflow
Upstream LSR, against " " "



Label Distribution

Unsolicited downstream

LSR generates Label as soon as it is ready to forward MPLS packets of the respective FEC

- Upstream Neighbours update forwarding tables

Label used as outgoing label

Downstream on demand

Downstream LSR generates label binding on demand

Upstream LSR has to request binding for FEC

Usually a MPLS-defined protocol was invented and used de-facto: RSVP-TE used mostly

RSVP (Resource-Reservation-Protocol)

goal: Resource reservation for end-to-end datastreams

establish connection and periodically indicate Liveness

they get removed, if no liveness is detected

Path Message

From sender to receiver

Find path to receiver

Every hop is recorded in the message

Resv Message

From receiver to sender

Bandwidth reservation on return path

RSVP-TE (+ traffic engineering)

Extension to support Label distribution

Many additional fields for functionality (fast reroute, ...)

Path Message
- from upstream LER to downstream LER
- Label request (downstream on demand)
- Bandwidth reservation

Resv Message
- response to Path Message
- Label binding (hop-to-hop)

VPNs implemented by label stacking
- Outer label identifies path to LER
- inner label " VPN instance / customer

Layer 3 VPN "BGP/MPLS IP VPN"

Customer may use private IP addresses
Different customers can have the same IP address

Provider takes part in routing
use of 64 bit "Route Distinguisher" to identify customer in BGP/MPLS
LER has Virtual Routing and Forwarding Table (VRF) for each customer

Label stacking
Inner MPLS Label mapped to VRF
Outer label defines provider, gives path through provider network
⇒ Different customers can share path as well as different paths for the same customer possible

Layer 3 VPN

Customer Edge Routers announce their subnets to their LER
LERs then signal this information to the other LERs
LERs " create FECs for announced subnet
Inner label also announced over BGP
outer " based on traffic engineering tunnels

Layer 2 VPN

Customer connects Ethernet switch to LER
Inner label then defines outport @ LER
No control traffic between customer and provider
In contrast to Layer-3 VPNs, where customer announces routes to provide

Label Distribution

Path Message	Backbone
Explicit Route	~~LER~~-KA1, Backbone-Z, Backbone-P, LER-P
Record-Route	LER-KA
Reservation	8014bit/s
Label	Request new label

↑ will move down for every Router passed!

Resv Message

Record-Route	LER-KA1, Backbone-KA, Backbone-Z, Backbone-P, LER-P
Label	2, 8, 7, 2 ← changed @ every Router that was passed by

6. SDN

Basic principles:

1. Separation between data & control plane
2. Control plane has global view
3. different modes of operation applyable
4. processing is based on flow
5. Network is soft-ware programmable

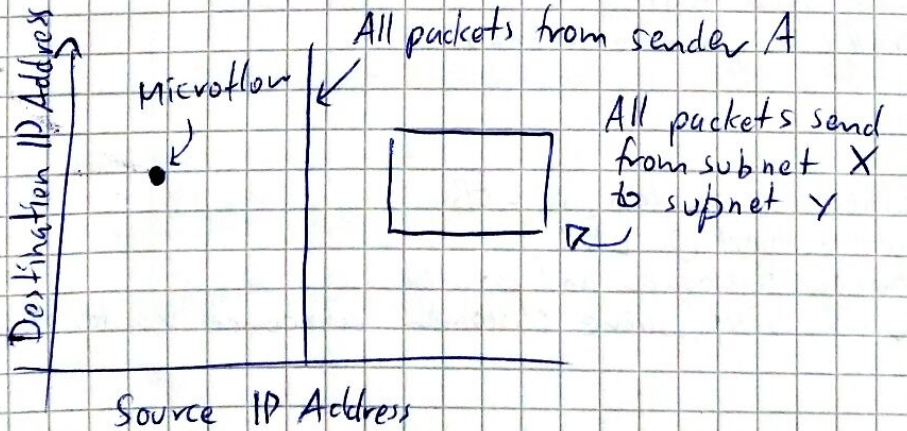
SDN Controller

generates, manages & sends rules for forwarding to SDN routers that they then store and obey.

Rule: <Match, Action, Switch>

- Match: ID for subset of a Flow
- Action: OP to perform
- Switch: Device on which it should be installed

Flow Space describes flows as objects in a multidimensional space



Global network view

A SDN Controller is logically centralized
assumes that it knows all switches and their configuration
as well as their topology
allows execution of link-state routing without protocol

Static configuration
Fixed settings prescribed by network operator
i.e. static forwarding policies
TCP connections between controller and switches, usually via static IPs

Active Monitoring
SDN controller is actively involved in refining the global view
i.e. periodic polling of information from the switches (for keepalive & port utilization monitoring)
Topology detection via Link Layer Discovery Protocol

Passive Monitoring
SDN Controller refines the global view by passive observation
i.e. Information learned from handshake w/ switch
" from internal state of switch (e.g. rules programmed so far)

Programmable via Software
via so-called network-apps
SDN controller can execute multiple apps in parallel
programmed via API (application programming interface)
SDN controller then generates rules <M, A, S>
1. can be proactive: @switch
operation is programmed before first affected package arrives
2. reactive: ask controller, if package arrives, for which no rule is known
the former is only feasable w/ flows that are large
doesn't have any additional delay
controller doesn't necessarily be online
the latter has a higher setup time and controller needs to be reachable
but allows fine-grained flows

SDN Architecture

Application plane	Network Apps that perform network control and management. Implemented against northbound API
Control plane	Control modules handle standard tasks, complexer ones are handed to the application plane. Interacts with the apps, switches, other SDN controllers and legacy infrastructures
Data Plane	Infrastructure for the packet forwarding / processing supports operations like: match, drop, forward

Open Flow (@Data Plane)

Specification for SDN-capable Switches
defined by Open Networking Foundation (ONF)
defines a common logical architecture for SDN-capable switches
rather the functionalities.
implementation stays vendor specific

also defines a standard for secure protocol for controller/switch interaction

OFS ≙ Open Flow Switch

Ports: Ingress / Egress = Input / Output
Physically a port / hardware interface
Logical abstractions possible for aggregation / tunneling

Port short-names: ALL ≙ all egress ports
IN_PORT ≙ references in-port of packet
CONTROLLER ≙ the interface to its controller
NORMAL ≙ back to vendor specific ports

Flow tables
Separate packet streams into flows
flow table entries identify specific flows
Match fields select packets by their header fields
priority determines the relative precedence of an entry

Match fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

this enables management functions like:
counting # processed packets
automatic removal of dead flow (life-time = 0)
set marker values (Cookies)
flags indicate how a flow is managed
Matches match packet data to corresponding match field values
Wildcards using bitmasks
empty match fields match all packets / flows
simultaneous matches can occur, highest priority then is taken.
Optional table-miss-flows
Matches all flows → enables reactive flow programming

Actions: forward, drop, modify header of a packet, push new tag onto a packet, remove a tag from packet

Open Flow Channel
Connects switch to a controller
provides southbound API functionality of an OFS
Management, configuration, event signals, monitoring (error states, liveliness, statistics ...) & experimentation possible
can have multiple channels to different controllers
3 types of messages:
1. Controller-to-Switch message (modify / read) (for new flow, port status change)
2. Asynchronous " (usually switch-to-controller)
3. Synchronous → " (hello / echo / Error)

Pipeline Processing
chaining of multiple flow tables
flow tables are numbered, no recursion possible

Action set
Ingress & Egress processing each have an associated action set
Accumulates actions during pipeline processing
actions are deferred until the end of processing unless told other

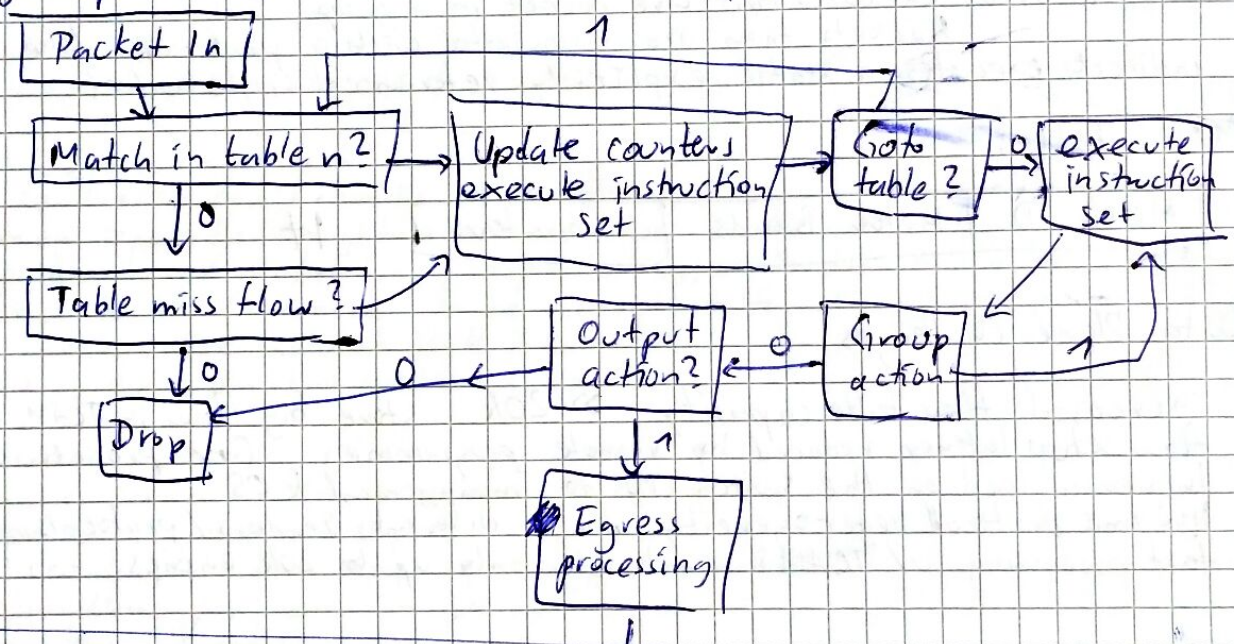
Instructions

control how packets are processed in the pipeline
instructions are more generic operations than actions
Flow table traversal can be controlled via go-to-statements
write/clear operations can be applied to the action set
or executed directly via apply instruction
Action sets can only have one " of each type. Therefore
older ones become overwritten
all instructions are then executed in a static order

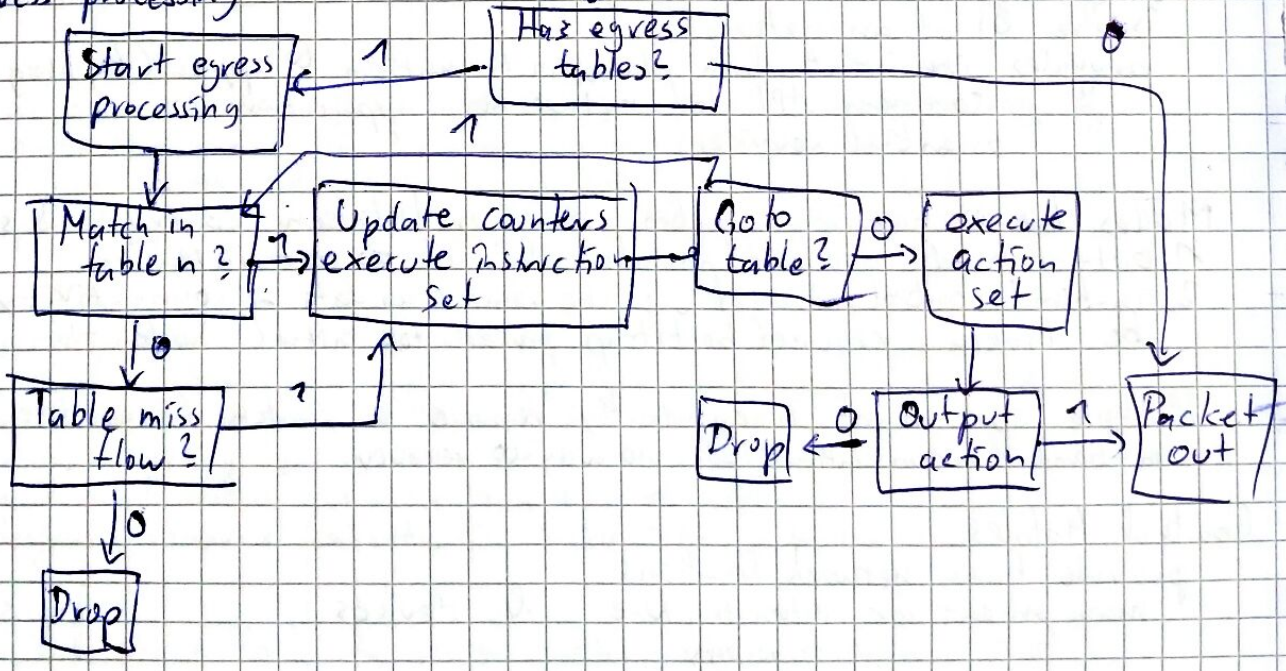
Learning switch

separate tables for learning & forwarding
Learning table matches source addresses and forward to controller, if
address was not yet learned
forwarding table matches destination " and forwards packet if
yet learned. If not, flood packet
Only 2-N rules for N hosts
but HW rarely supports two tables

Ingress processing: table 0, action set empty



Egress processing



Group tables

Group Identifier	Group type	Counters	Action buckets

represent additional forwarding methods
e.g. fast failover, Link selection, ...
can be created by ingress processing
optionally groups may perform group actions (Group chaining)
effect depends on group type and their action buckets

Action Buckets

each group has ≥ 0 group buckets
not every action bucket needs to be executed
a group w/o action buckets drops a packet
action buckets contain a set of actions to execute, just like an action set

Group types have specific semantics

all: execute all buckets in a group, selected by method (hash, round-robin, ...)
select: selects only one of many buckets in a group
fast failover: executes first live bucket in a group
Buckets than are associated with a port, up = live
indirect: executes a static, explicitly referenced single bucket

Meter tables

Meter ID	Meter Bands	Counters

Data Plane challenges

Increased flow table capacity >> 20k store big flows in TCAM, others in RAM
flow setup latency reduced by remote programming (use proactivity)
Increased load on the switch CPU for management & OS
HW and protocol heterogeneity due to different vendors / protocols
fast processing w/ TCAMS can handle only up to 20k entries

SDN Control Plane

is the OS of an network
provides consistent data plane information to apps (topology, flows, stats...
" common API for networking applications
" essential services

Modes for exchanging control messages between controller & switch:
1. out-of-band: dedicated physical channel ⇒ cost intensive
2. in-band: control messages use the same " as data ⇒ connectivity can be broken, requires bootstrap process for initial switch-to-controller

Multiple apps might independently remove a switch from the system
in-band communication lost, no way to recover

Control Modules

provide basic network functions
management for infrastructure & devices
topology
statistics

Shortest path forwarding
isolation and security enforced between apps & services
delivery of notifications between SDN switches and apps
App use these control plane functions to realize/implement complex services

Northbound control plane functions to applications
abstraction from underlying hardware

Service Abstraction Layer
Provides basic services to controller functions and apps
Southbound interfaces are treated as plugins
Maps service requests to the most appropriate southbound protocol
⇒ provides uniform access to the underlying network

Plugin Manager
exposes functionality in form of abstract features
i.e. change source or destination fields of IPv4 header
decrease TTL in IPv4 header
configure a queue

Service Manager
maps service requsts to feature requests

Distributed Controllers
1 controller can't handle ∞ switches
domain separation might also need more controllers
⇒ problems w/ scalability, responsibility, reliability, incremental deployment
but network view needs to be consistent for all controllers
⇒ synchronize network state information
this is done via the westbound interface

Westbound Interface
enables communication between SDN Controllers
for data import & export
coordinated flow setup, exchange of reachability information
algorithms for data consistency
monitoring & notification
can support heterogeneity between controllers

Synchronization
Controller directly apply internal operation & notifies remote controllers of relevant changes in the network
Apps can perform data plane operations on remote switches
operations are then delegated to responsible SDN controller

Abstractions
apps only see an abstracted network
Big switch abstraction, for some apps that doesn't need to know of long router chain, it can be omitted
Network slicing as a form of virtualization
individual users get their own slice of network
requires resource orchestration inside the controller

Challenges
Scalability: logically centralization requires strong HW
control plane can be overloaded by data plane site

Important parameters w/ scalability implications
- number of remotely controlled switches
- number of hosts / flows in the network
- " " messages processed by the controller
- communication delay between switches & "

possible solutions:
Distributed Controllers
parallelism-based optimization such as multi-threading, batch processing
optimized routing schemes that consider flow table size / amount of rules

Challenge	Possible Solution	Tradeoff
Comm. delay, flow setup latency	Proactive mode of Separation, wild cards	Microflows may become invisible
Limited control flow table capacity	Caching mechanisms	additional HW Cache miss increases WCET of flow
High number of controller events	distributed controller architectures	problems w/ Consistency & synchronization

Consistency
multiple controllers inherit the problems of a distributed system
CAP theorem: Consistency: system responds equally, no matter which node responded
Availability: system is always responding
Partition tolerance: system continues to function even if messages are lost or parts of network fail
It is impossible to provide all 3 @ same time - Maximum of 2

SDN Applications
Network is software programmable, Apps provide functionality
functionality depends on the environment like: data centers, enterprise network, home network, IXP
Broad classification into 5 wide areas:
TE
Mobility & Wireless
Measurement & Monitoring
Data Center Networking
Security & Dependability

Load balancing in SDN
reactive: load balancing algo triggered for every new connection (Microflow)
needed apps: Flow Manager, net manager (monitoring usage)
Host " (monitoring server liveliness)
proactive: preinstall forwarding rules using address space partitioning
using wildcards to reduce # rules
each server gets a weight α_i
use binary trees to model address space
normalize weights, so that $\sum \alpha_i = 2^n$, $n \in \mathbb{N}$
Map nodes to leaves according to α_i
optimizations like aggregating sibling nodes

Traffic dynamics
traffic changes over time. No problem for reactive solution.
proactive solution requires transition strategies
temporarily intercept all connections and redirect to controller

Policy Cop for TE
optimize resource utilization
apply (customer specific) policies
QoS with regard to SLAs
Validator monitors network to detect policy violations
Enforcer adapts rules based on network conditions and high level policies
Adaption actions:
packet loss → reconfigure queues or reroute to better path
throughput → " rate limiters to throttle misbehaving flows
Latency → reroute to path w/ less congestion & suitable delay
Jitter → " " " " "
Device failure → " " " that bypasses the failure

7. Network Function Virtualization

Middlebox ≙ Device on path between client & server that performs other tasks than forwarding / routing
NAT, firewall, proxy, load balancing, intrusion detection

Network Address Translation
allow inside a realm globally non-unique IP-Addresses
can't be used for routing
provides mechanism to connect a realm w/ private addresses to an external realm w/ globally unique IP Addresses
Exchange globally unique & private addresses when packets traverse network boundaries

Recalculate checksum
Collisions on port numbers need to be resolved
as well as state of every data stream needs to be traced

Firewall
establishes barrier between trusted & untrusted network
drops or forwards packets based on ruleset
shallow: only check header of layer 3, 4
deep inspection: look @ data input
stateless: every packet is inspected independently
stateful: remember state of connections (like seq numbers)

Proxy
Intermediate device that acts like server & client
performs requests based on requests of clients
appears to the client as a server and to the server as a client
Transparent: doesn't change data / requests
Non-" : modify requests / responses in order to provide additional services
Web caching: Proxy stores responses from server, accessing the same content again is faster, if proxy is closer than the server to the client

Caching used by content delivery providers to ensure audience has low latencies when accessing content. Placement of cached content depends on size of audience in an area.
Middleboxes often built as proprietary hardware
fast, but inflexible & black box for infrastructure operator
Static wiring: hard to setup/teardown/improve/move

Network Function Virtualization (NFV)

Mimic ideas of cloud-computing
Network functions are implemented in Software
and therefore decoupled from the hardware they run on
⇒ can be executed on shared HW
⇒ can be easily moved and instantiated everywhere
Servers are often close to network nodes or to end user premises

Network Service

combine multiple network functions
end-to-end behavior of a network service is the combination of the individual network functions passed by on a path

Benefits:
resource sharing
Agility & flexibility } reduced costs, no special personnel, x86
rapid deployment

NFV Infrastructure (NFVI)

can represent more Points of Presence (PoP)
many enterprises can be in the same NFVI
redirected transparently by SDN
can be done w/ MPLS
each content delivery provider can then easily choose where to open new PoP

Virtualization

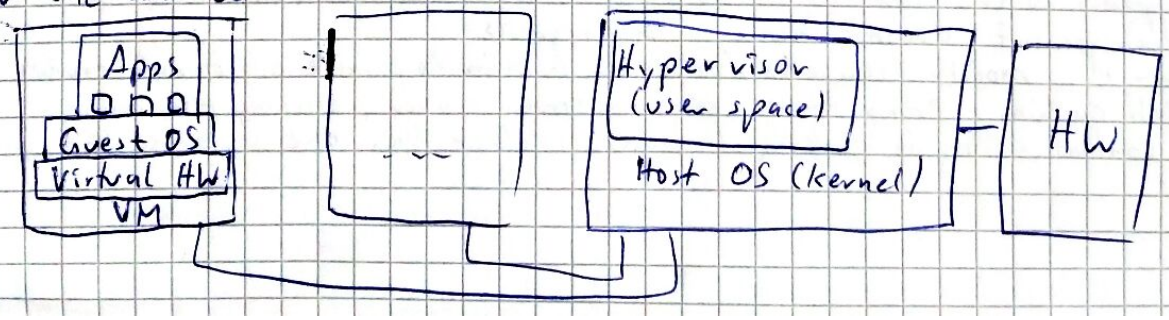
abstraction layer between HW and OS on VM, often called hypervisor
acts as a resource broker between HW and VMs
translates between I/O from VM to shared HW
through isolation, multiple coexisting VM instances are possible
allows live migration of VM to other host

Type-1

runs directly on HW
high-performance
strong isolation
synchronizes access of VM to the HW

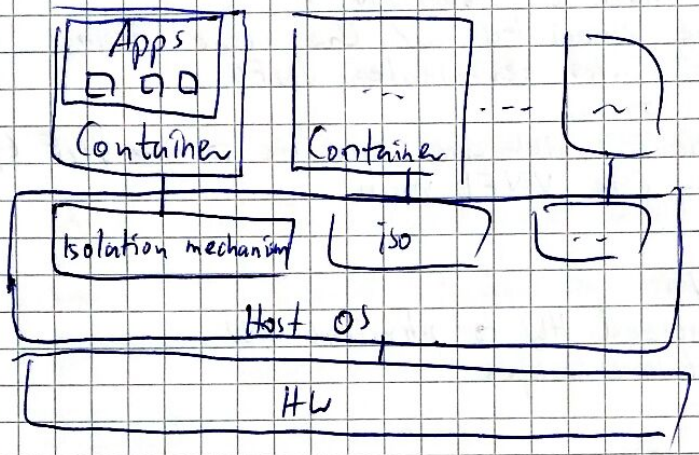
Type-2

runs on an hypervisor OS
executed as an app in user space
interaction of VM and HW is directed to physical devices through a VM-driver or the host OS



Container based virtualization

single kernel provides multiple instances (containers) of the same host operating system
- no hypervisor involved
- isolation enforced by host OS
  - each container has its own view of OS
- Apps in containers are executed by the host OS
  → apps depend on it
- Kernel synchronizes access of containers to HW



High level view on NFV-Framework

3 main building Blocks:

VNF: the actual functions provided by the software

NFVI: HW & SW used to execute VNFs

NFVO: Orchestration of network service management & NFVI resources (MANO)

Functionality (largely) the same as non-VNF networks, but implementation is done in software
Scalable by adding virtualizable HW resources
implementation HW independent

NFVI ≙ sum of HW and SW components that build up the environment on which NVF apps run
provides physical platform
manages computing, storage & network resources
usually based on low-cost standard HW

MANO ≙ Management & Orchestration
responsible for MANO of NFV resources
includes computing, networking, storage, VMs
all virtualization specific tasks
consists of 3 parts:
1. NFV Orchestrator
2. NFV Managers
3. Virtualized Infrastructure Manager

NFV Orchestrator
performs lifecycle man. of network services
man. and coordinates NFVI resources across multiple virtualized infrastructure man.
may instantiate VNF man.
handles topology between them
policy man.

NFV Manager
responsible for lifecycle man. of VNFs
multiple VNFs can be associated w/ the same man.
instantiates, configures, scales, terminates VNFs
collects statistics, error logs...
can also provide functions which are specific to a VNF type
requests resources from the VNFI man.

NFVI Manager
man. resources of NFVI
man. mapping of virtualized HW to physical HW
runs hypervisors / software images

Service Function Chain (SFC)
ordered set of network functions
allows creation of composed network services
like advanced caching: authentification → firewall → cache
done by label stack
represented by a forwarding graph (may be cyclic)

Challenges

Security
Performance 100+ GBit/s
placement
reliability
testing & debbuging
carrier grade requirements
existence of legacy networks

8. Ethernet Evolution

Aloha
senden, drahtlose Übertragung
wahlfreies senden, Kollisionen möglich
Asynchroner Zugriff

Kollisionserkennung
Explizit: Bestätigung durch höhere Schichten
separater Kommunikationskanal erforderlich
Implizit: spiegele empfangenes Paket, Sender vergleicht dies
Reaktion: Sendewiederholung nach zufälligem Zeitintervall
Abbruch nach zu vielen Sendewiederholungen

Slotted Aloha
synchroner Zugriff in Zeitschlitzen
im Mittel weniger Kollisionen
Eingesetzt bei GSM (Steuerkanal)

Metrik für Kommunikationssysteme $U_{max} \hat{=}$ maximale Auslastung
wie gut kann das Medium ausgenutzt werden?

hierzu braucht man viele Variablen:

r	Datenrate der Übertragungsstrecke [bit/s]
r_e	" erzielt [bit/s]
N	#teilnehmender Systeme
X	zu übertragende Datenmenge [bit]
t_w	Wartezeit, bis das Medium zugeteilt wird [s]
t_s	Sendezeit [s]
U	Auslastung
d	maximale Distanz zwischen zwei Systemen [m]
t_d	Ausbreitungsgeschwindigkeit [m/s]
a	t_a / t_s
p	Wahrscheinlichkeit für Datenübertragungswiederholung

Slotted Aloha W'keit für geglückte Übertragung: $Np(1-p)^{N-1}$

$\Rightarrow U_{max} = 1/e = 0{,}36 = 36\%$

Aloha W'keit für geglückte Übertragung: $Np(1-p)^{2(N-1)}$

$\Rightarrow U_{max} = 1/2e = 0{,}18 = 18\%$

CSMA basierte Verfahren (Carrier select multiple access)

Medienzuteilung: Zeitmultiplex, variabel, zu fälliger Zugriff
vor Senden prüfen, ob Medium belegt (Listen before talk)
Asynchroner Zugriff

CD (Collision Detection) Ethernet
Listen before and while talk

CA (Collision Avoidance) WLAN

Ethernet

Zeitmultiplex, variabel, zufälliger asynchroner Zugriff
Kollisionserkennung durch Mithören an der sendenden Station
Exponentieller Backoff
1-persistent
96 bit keine Aktivität → frei
Senden ist 1-persistent
Jamming Signal: 48 bit Präambel
Exponentieller Backoff für Sendewiederholungen
Kollisionserkennung muss vor Beendigung des Sendevorgangs erfolgen, ansonsten reines CSMA
2 * Ausbreitungszeit ≤ Sendezeit
Erreichbar durch das Padding-Field PAD

Ethernet-Rahmen

PR (7)	SD (1)	DA (6)	SA (6)	Type (2)	Data (≤ 1500)	PAD	FCS (4)

Präambel für Synchronisierung (10101010...)

Start of Frame Delimiter zeigt Beginn an (10101011)

DA Zieladresse

SA Sourceadresse

Type/Length des darüberliegenden Protokolls oder Länge der Nutzdaten

Datenfeld

PAD Paddingfeld (optional)

FCS Frame check sequence

Zwischen Frames liegt ein Inter Frame Space (IFS) von 96 bit

Beurteilung der Auslastung

$$r_e = \frac{x}{t_s + t_a}$$

$$\Rightarrow U_{max} = \frac{1}{1+a}$$

$$\lim_{N \to \infty} U_{max} = \frac{1}{1+3{,}44a}$$

Fast-Ethernet

10 Mbit/s → 100 Mbit/s
automatische Erkennung & Umstellung durch Protokolle
Netztopologie: Stern
Halbduplex oder duplex Links

CSMA/CD
Codierung: 4B/5B, NRZI

Stau an Switches

Workarounds mit Backpressure
- Kollision herbeiführen
  - => Sender bricht ab und wiederholt Rahmen (Backoff)
  - => zu viele Wiederholungen → Verbindung beendet
- Medium als belegt vortäuschen (Präambel senden)
  - → Sender wartet auf Slot

=> Implizite Flusskontrolle

Duplex Links ohne CSMA/CD brauchen anderes Workaround
=> explizite Flusskontrolle:
PAUSE-Funktion
Sende PAUSE-Rahmen mit Zeitangabe
PAUSE mit Zeit = 0 ≙ explizites "GO"
Bremst keine darüberliegenden Protokolle, löst aber auch nicht langfristige Überlastungen im Netz

Control teil von
PAUSE-Funktion im MAC-Layer 2

MAC-Kontrol Frame

PR(7)	SD(1)	DA(6)	SA(6)	Type (2)	MCC(2)	Parameters (44)	FCS(4)

Type = 0x8808

MCC = MAC Control Opcode

0x0001 ≙ PAUSE

Kann nur bei Duplex-Betrieb genutzt werden
wird an vordefinierte Multicast-Adresse geschickt
Parameter: gewünschte Auszeit
fülle mit 0en auf

Automatische Konfiguration

Ziel: Geräte erkennen selber, mit welcher Datenrate & Kodierung eine Kommunikation möglich ist.
Zwischen Endsystemen und Knoten (Hub, Switch)
während der Initialisierung
Austausch von 16 bit langen Nachrichten signalisieren mögliche Konfigurationen, als Folge von Takt- und Datenpulsen gesendet

Gigabit/s-Ethernet

Geänderte Kodierungsverfahren:
8B/10B mit NRZ
Datenrate 1000 Mbit/s (abwärtskompatibel zu 10 Mbit/s und 100 Mbit/s)
Problematisch: Kollisionserkennung bei halb duplex-Links

Carrier Extension
soll Kollisionserkennung sicherstellen
Künstliche Erhöhung der Übertragungszeit ohne die minimale Länge der Rahmen zu erhöhen

Grundlegende Änderung:

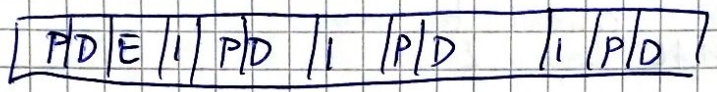
Länge des Zeitschlitzes ≠ Länge der minimalen Übertragungsdauer

512 B 512 b

⇒ Carrier Extension zwischen 1 und 448 b lang
" " wird physikalisch als non-data-bits gesendet

Frame Bursting
Ziel: effiziente Übertragung kurzer Rahmen
Stationen dürfen einen Burst von Rahmen direkt hintereinander senden
Erster Rahmen mit Carrier Extension, falls notwendig für die CD
Letzter " muss nach mindestens 8192 B beginnen

Schema



P ≙ Präambel & Start of Frame delimiter
D ≙ Daten
E ≙ Carrier Extension
I ≙ Inter Frame Space

10 Gbit/s - Ethernet
Glasfaser / twisted pair
Datenrate: 10 Gbit/s
Nur Punkt-zu-punkt Verbindungen
Ausschließlich Vollduplex-Betrieb
kein halbduplex-Betrieb mehr, keine Hubs
kein CSMA/CD
kein Frameburstring
keine Carrier Extension

Weiterhin: 96 b IFS
1518 B maximale Größe eines Rahmens
64 B minimale " " "
Zusätzlich: Jumbo-Frames: 9014 B Nutzdaten

Improved coding mechanisms:
Carry multiple bits per transmission step: 16-level pulse amplitude m
lation

Interference becomes more problematic:
Near end ~~or~~ crosstalk: NEXT
Far " " : FEXT
Crosstalk from neighbouring cables: Alien Crosstalk: AXTLK
Signal-to-noise ration SZNR critical

Not a typical solution for a network @ workplace / home
rather connecting buildings 10G Base-SR (400m) } 64B/66B Code
10G Base-ER (40km)
10G Base-T (100m)

40 / 100 Gbit/s – Ethernet

Distances up to 40km
multiple parallel channels
twisted pairs / different frequency / wavelengths
Virtual Lanes contain data streams
aggregate different data streams into one virtual channel and enclose them with identifier (64B/66B → 46B/66B)
apply bit-level multiplexing, if #channels < # virtual channels

⇒ Speeds have increased by x10k
variety of different medias used
extends to WAN
switched Ethernet is dominant
MAC-Protocol often unnecessary
One constant remained:
Ethernet frame format

Spanning Tree

Brücken: Ziel: Kopplung von LANs auf Ebene 2
Filterfunktion: Trennt Intranetzverkehr von Internetz-Verkehr
Source-Routing Brücken:
Endsystem fügt Wegewahl-Info in Paket ein
Brücken leiten es anhand dessen weiter
Senden über Brücke nicht transparent
Transparente Brücken:
Lokale Weiterleitungsentscheidung an der Brücke
Wegewahlinfo in Tabelle enthalten
Statisch / gelernt
Endsystem muss nichts von der Brücke ~~Lernen~~ wissen

	Transparente Brücken	Source-Routing-Brücken
Transparenz	✓	✗
Verzögerung	gering	ggf. hoch
Routing	nicht optimal	optimal
Alternative Wege	✗	✓
Kapazitätsausnutzung	Schlecht	Gut
Skalierbarkeit	Schlecht	Gut
Reihenfolgeerhaltung	✓	✗

Transparente Brücken ≙ Switch
hat für jeden Netzanschluss eine eigene Schicht-1 und MAC-Instanz
Datenpfad über MAC-Relay
Implementiert per Weiterleitungsfunktion auf Schicht-2
Kontrollpfad geht durch LLC (Logic Link Control) höher an Schicht-3

Aufgaben: etablieren einer schleifenfreien Topologie
Weiterleiten von Paketen durch lernen der Lokation von Endsystemen
Aufbau der Filterdatenbasis

Spanning Tree Algorithmus (Brücken protokoll)
Organisierung der Brücken und Links zu einer Baumtopologie
Weiterleitung nur am Baum entlang möglich

Voraussetzungen:
MAC-Adresse aller Teilnehmer eindeutig
Kennung " " "
Anschlusskennung " " "
Pfadkosten aller Anschlüsse bekannt

Brücken versenden spezielle Pakete
Bridge Protocol Data Units (BPDU)
beinhalten Kennung der sendenden Brücke
" " angenommen Root-Brücke
Pfadkosten von sendender - zu Rootbrücke

Algo:
1. Bestimmen der Rootbrücke
2. " " Root-Anschlüsse jeder Brücke (Σ wes zur Rootbrücke)
3. " " Designated-Brücke für jedes LAN
LAN kann mehrere Brücken konfiguriert haben
Geringste Kosten am Root-Anschluss $\triangleq$ designated bridge
bei gleichen Kosten regelt die Brückenkennung
4. Schalte alle anderen Brücken aus

Initiell haben alle Brücken keine Topologie-Infos
$\forall$ Brücke: Annahme: ich bin die Rootbrücke
sende periodisch BPDUs
Bei Empfang einer BPDU mit kleinerer Kennung ⇒ nimm nicht weiter an, selber Rootbrücke zu sein und sende keine BPDUs mehr
Bei Empfang weiterer BPDUs, ggfs. Update der Konfigurationsdaten.
Kleinere Kennung der Rootbrücke → Update
gleiche " aber kleinere Kosten zur Root → Update
gleiche Kennung der Rootbrücke & kosten gleich aber sendende Brücke hat kleinere Kennung → Update
Brücke stellt fest, dass sie nicht designated-bridge ist → keine Weiterleitung mehr
Rootbrücke sendet periodisch BPDUs weiter
nur aktive Bridges leiten diese weiter
Ausbleiben von BPDUs ⇒ Brücke wieder Root und wiederhole
Nach Stabilisierung werden Pakete über die entsprechenden Ports weitergeleitet entsprechend der Einträge in der Filterdatenbasis

Filterdatenbasis.
Enthält für die Weiterleitung erforderliche Daten:
Zieladresse, Ausgangsport, Zeitgeber
Statische und dynamische Einträge
vom Sysadmin erstellt
gelernt von durchlaufenen Paketen
verlernt durch Zeitgeber
Pakete mit lokalem Ziel werden nicht über die Brücke weitergeleitet

Empfang eines Pakets
Ziel bekannt?
Identifiziert durch Ziel-MAC-Adresse
Weiterleiten an Port gemäß Tabelle
Unbekannt?
An alle anderen Ausgänge fluten
Quellsystem lernen, ist über Eingangsport erreichbar
Neuen Tabelleneintrag erstellen

Rapid Spanning Tree

Neue Zustände der Ports
Alternate Port: bester alternativer Port zur Root-Brücke
Backup-Port: alternativer Port zu einem Netz

BPDUs werden nun als keep-alive-signal verwendet
⇒ Erkennung eines Ausfalls schneller möglich

Neue Typen von Ports:
Edge-Port: nur Endsysteme (⇒ schleifenfrei)
Point-to-Point: Nachbar ist eine Brücke, Vollduplex
Shared Port: Verbinden zu einem Netz

Echtzeit Ethernet
Für bspw. Windräder, Fahrzeuge
Unterstützt Echtzeitfähigkeit
Kompatibel zu Standard Ethernet
„Time-Triggered-Ethernet" TTE

3 Versandsklassen
Zeitgesteuerte, hochpriore Rahmen (TT) Feste, globale Zeitschlitze
Ratenbasierte Datenströme (rate-constrained) Maximale Größe, Mindest Abstand
Best-Effort-Verkehr (u.a. Standard Ethernet)

TT-Ethernet-Rahmen
in der Empfängeradress wird die Nachrichtenart kodiert
Critical Traffic (CT) Marker & CT-ID im Header
Entweder TT oder rate consbrained
Erkennung von CT über Bitmaskenvergleich
Verwenden für zeitplankonformes Scheduling und Weiterleitung von time-triggered Paketen

Ether Cat
Ethernet for Control Automation
Steuerungsaufgaben auf Feld-Bus-Ebene
Effizienter Umgang mit kleinen Datenmengen (8-bit Messwert)
Niedrige Latenz & Jitter für Roboterautomation

Audio Video Bridging (AVB)
Für Rundfunk, Veranstaltungstechnik (Konzert, ...)
Zeitsynchronisation zwischen AVB-Stationen < 1ms
Sehr exakte Echtzeituhren
Jitter < 100ns
Einsatz von Hardware-Zeitstempeln beim Senden & Empfangen

9 Data Center Networking

Typical Services:
IaaS: Infrastructure as a service (for virtual computing, ...)
PaaS: Platform " " " (" Web development, ...)
SaaS: Software " " " (" Servies like Dropbox...)
StaaS: Storage " " " (Storage of a cloud provider)

Data Center 100k +
typically has large # of compute servers and VMs
switches w/ small buffers
Should be extensible w/o massive reorganization
Needs to be reliable (redundancy...)
" " " highly performant (Load of 100 Tbit/s t @ Low Latency)

Top-of-rack-Switches (ToR)
connects servers within rack
48 x GBit/s Ethernet w/ 10 Gbit uplink
small buffers
42-48 rack units / rack
Challenges:
Maintain scalability 100k + - 1m servers
maximizing throughput while minimizing latency & cost
guarantee integrity & service availability
enhance power efficiency

Data Center Network
Interconnects data servers with each other
as well as them to the internet (border routers)
⇒ two types of traffic: external clients <-> servers
servers <-> servers
Commodity protocols:
TCP/IP
Ethernet
Routing / Forwarding within data center
needed efficient way to communicate between two servers
⇒ " utilization
avoid forwarding loops
detect failures quickly
provide flexible and efficient migration of VMs between servers
no change of IP Address to keep TCP connections opened

Layer 3 routing? OSPF?
routing tables are too slow
VMs moved require new IP address
manual configuration

	Plug and play?	Scalability?	Small Switch State?	Seamless VM Migration?
Layer 2 MAC	✓	✗	✗	✓
Layer 3 IP	✗	✓	✓	✗

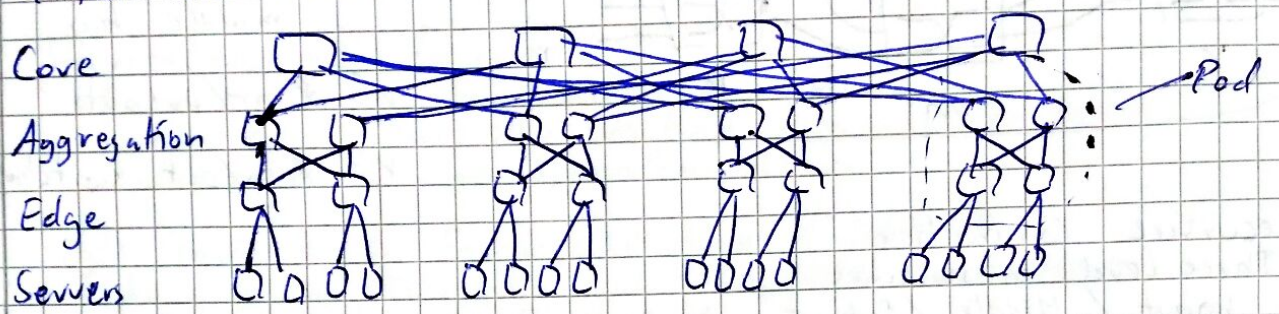
Why not Ethernet?

Internet Tree Topology:
East-West-Traffic: Between internal servers and server racks
North-South-" : Result of external request

Fat Tree Networks

Problem: Connect large number of devices using switches that only have a limited # ports
Characteristics: For any switch: # Links going down to its children is equal to # links going up to its parents.
⇒ the link gets fatter to the top of the tree
But switches w/ high number of ports are expensive

4 Pod Fat Tree



K Pod fat tree

Each switch: k ports
Up to $k^3/4$ servers using $\frac{5k^2}{4}$ switches and $k^3/2$ Links
Edge and aggregation arranged in k-pods
$k/2$ edge switches and $k/2$ aggregation switches per pod
Edge " can connect to $k/2$ servers
Core " : $k^2/4$, each has hon connected to each pod

⇒ Special case of Clos-Network

Private address vorom: 10.0.0.0/8
use 10.pod.switch.1
where switch is enumerator for edges first, then ($k/2$ +)
for aggregation

End hosts are addressed: 10.pod.switch.ID where ID is enumerator "from left to right" starting w/ 2

Also via Layer 2 forwarding possible

Advantages of k pod fat tree:
All switches are identical, cheap ones can be used
multiple paths w/ equal cost
Disadvantage: high cabling complexity

Load Balancer

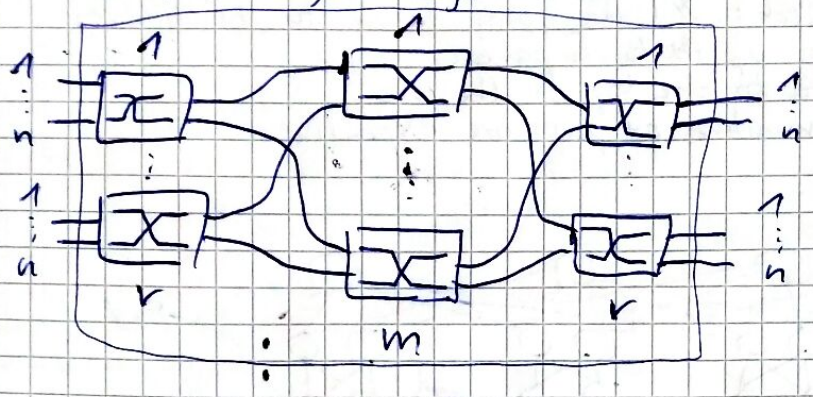
typically on per application
distributes load to servers
that server might delegate that traffic/load to another server
collects info of traffic from Servers
can do NAT
hides internal net structure

Hierarchical Topology

supports scalability
redundancy ⇒ multiple point of failure ⇒ reliability, availability

Clos Network

Multi routed tree
allows forming a large switch w/ smaller ones



Referred to as an
(m, n, r)-switch

$m \hat{=}$ # switches in the middle stage

$n \hat{=}$ # in-/outputs

$r \hat{=}$ # in/out switches

Recursive Construction
Three levels of switches
Input / Middle / Output

Clos Network are not limited to these 3
replace Middle switches by 3-Level Clos Network

Non-blocking @ $m \geq 2n-1$

Input switches → ToR @ Servers
Middle " → Aggregation @ ToR
Output " → Intermediate @ Aggregation

Ethernet @ Data Centers

Ziel Vereinheitlichung für SANs (storage area network)

Aber Ethernet bietet Best-Effort-Dienst
keine Garantien
Variablen Verzögerungen
Spanning-Tree zwar schleifenfrei
aber nicht flexibel, keine optimalen Pfade
⇒ Unterauslastung der Netztopologie
Priorisierung mit Q-Header
→ TPID (Tag protocol identifier), 16 bit
→ TCI (Tag control information)
3b Priorität
1b Drop eligible Indicator
12b VLAN ID

Data Center Bridging

Vereinheitlichte, auf Ethernet basierende Lösung für unterschiedliche Anwendungen in Datenzentren
Erweiterungen zu Ethernet:
Prioritätsbasierte Flusskontrolle
Enhanced Transmission Selection
Staukontrolle auf Schicht 2
Data Center Bridge exchange

Prioritätsbasierte Flusskontrolle
⇒ Datenverluste durch Stausituationen vermeiden
Erweiterung des PAUSE Flags um 8 Prioritäten
Differenzierter QoS möglich

Enhanced Transmission selection
Reservierung von Bandbreite durch Einführung von Prioritätsgruppen
kann mehrere Prioritätsstufen eines Verkehrstyps enthalten
Garantie einer minimalen Datenrate
Rest kann von anderen Prioritätsgruppen genutzt werden

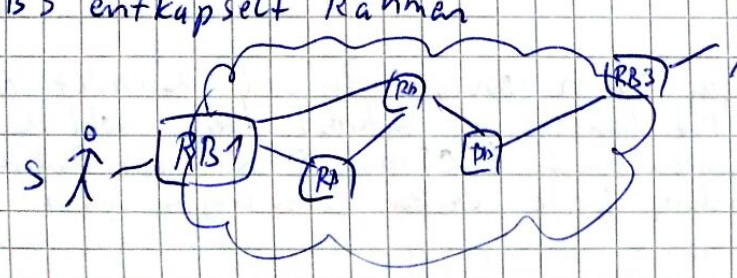
Staukontrolle auf Layer-2
Quantized Congestion Notification (QCN)
Versendet auf Basis der Warteschlangenlänge im Buffer
an Verursacher per speziellem Rahmen

Data Center Bridge Exchange Protocol (DCBX)
Erkennung der Fähigkeiten und Konfiguration der Nachbarn
z.B. prioritätsbasierte Flusskontrolle
periodische Broadcasts an alle Nachbarn

Alternativen zum Spanning Tree
Ziele: Mehr Flexibilität & Ressourcennutzung
bei Topologieänderung, Lastbalancierung, Fast-Recovery
und zur Nutzung besserer Pfade
Skalierbarkeit für Netze mit > 1000 Brücken
Einsatz des Link-State-Routings, modifiziertes IS-IS
Basiert im Gegensatz zu OSPF nicht auf IP-Adressen
keine manuelle Konfiguration
Unterstützung von multipath
Nutzung von Tunneling
Kapselung der Rahmen innerhalb einer Domäne
Hinzufügen eines neuen Headers für die Weiterleitung

Shortest Path Bridging
Jede Brücke berechnet kürzeste Wege im LAN
Pfade müssen symmetrisch sein
Lernen die MAC-Adressen
Unterstützt Equal Cost Multigraph
Gleiche Pfade für Unicast & Multicast
MAC-Adresstabellen zur Berechnung kürzesten Pfade genutzt
Ziel unbekannt? → erstmal in kürzesten Pfad weiterleiten.
Pakete werden im LAN gekapselt (MAC in MAC)

Transparent Interconnection of Lots of Links (TRILL)
Neuer Header für Einkapselung
Routing Bridges implementieren TRILL
RB1 Kapselt Rahmen von S
und gibt RB3 als Ziel an, dort befindet sich D
RB3 entkapselt Rahmen



Jede RB berechnet kürzeste Wege zu allen RBs
Berechnung mehrerer Bäume möglich
Bäume für Multidestination genutzt
Weiterleiten, falls Destination unbekannt
⇒ Kleine Forwarding Tabler, keine Endsysteme
Header:

Outer Ethernet	TRILL	Inner Ethernet

- Outer Ethernet: MAC-Adress für Punkt-zu-Punkt Weiterleitung. Ändert sich auf jedem Hop
- TRILL: Nickname der Ingress Bridge und Egress; Multidestination Flag, Hop Count
- Inner Ethernet: Quell- und Ziel MAC-Adresse der Endsysteme

TCP within Datacenters

Wichtiges Hintergrundwissen:
oft sehr kurze RTTs, Server stehen dicht beieinander (µs statt ms)
Incast: oft many-to-one Übertragung
Multi-Paths, es existieren redundante Pfade im Datacenter
Lange & kurze Flows gemischt (Suchanfrage vs. Backup)
Virtualisierungen beeinflusst RTT
Ethernet bereits für Stau- und Flusskontrolle
Commodity Switches mit kleinen Buffern (sogar geteilt unter Ports)

Ziele
Gleichzeitige Unterstützung von geringen Verzögerungen und hohen Durchsatz
Hohe Auslastung der Resourcen / Infrastruktur
Geringe Kosten

Möglichkeiten
Datacenter hat eine administrative Kontrolle
Meist homogene Komponenten
Backward compatibility
Interner & externer Verkehr gut separierbar

TCP-Reno
benötigt große Buffer
für große Ende-zu-Ende Verbindungen geeignet
Verbesserte ECN und RED ⇒ DCTCP (Data Center TCP)

TCP Incast-Problem
häufig, das mehrere benachbarte Server quasi synchron antworten
race-condition, Buffer voll
kann auch Flows betreffen, die gar nicht zu den Incast verur-sacht haben. Da Buffer geteilt werden (unter Ports)
⇒ Retransmission Timeouts
⇒ Incast Collapse

Barrier Synchronization
Client hat TCP Verbindungen zu n Servern offen und versendet an alle n gleichzeitig eine Anfrage. Die langsamste Antwort bleibt wohl aus - Buffer voll. Wenke auf Retransmission timer (200ms ≫ RTT im Datencenter) Zeit, in der TCP-Verbindung keine Daten übertragen kann
⇒ Anwendung blockiert

Verbesserung durch Verringerung des Retransmission timers
↳ RTT im Data center
Desynchronisierung für retransmissions, sonst passiert das gleiche wieder
⇒ Randomisierung timeout = (RTO + (rand(0,5) + RTO)) $2^{backoff}$

Trotz niedriger RTT werden hohe Latenzen beobachtet
Ursachen:
Switches mit gemeinsamen Puffern für mehrere Ports
Incast (kürzere timeouts können helfen)
Abbau der Queue
Langlebige Verbindungen füllen die Buffer
kann zu synchronem Paketverlust führen
TCP-fairness hilft auch nicht...

Data Center TCP (DCTCP)

Ziel: mit commodity switches hohe Bursttoleranz, geringe Latenzen und hohen Durchsatz erreichen

Eigenschaft: DCTCP arbeitet mit geringem Füllstand der Queues ohne den Durchsatz zu reduzieren
Umgesetzt durch Reaktion auf Stärke eines Staus, nicht auf die alleinige Präsenz.

ECN-Markierung und RED Schwellwert basieren auf Füllstand der Queue, werden ab Position k gesetzt

DCTCP setzt das ECN-Echo-Flag CE nur bei Quittierungen für markierten Paketen

Staukontrolle beim Sender
Schätzung für den Anteil α der mit ECN markierten Segmente

$\alpha = (1-g)\alpha + gF, \quad 0 < g < 1$ F = Anteil markiert im letzten Fenster

Für jedes Fenster von Segmenten neu berechnet
α schätzt Wkeit, dass Länge der Queue > k
⇒ reagiert auf Stärke des Staus
Verfahre nach AIMD, falls keine Segmente markiert
$Cwnd = (1-\alpha/2)Cwnd$ sonst

Vorteile von DCTCP:

Incast, falls sich die Queues über mehrere RTTs aufbauen, hilft frühes Reagieren von DCTCP
Aufbau einer Warteschlange wird bei k Segmenten gestoppt
behandelt ⇒ mehr Buffer für Bursts vorhanden
Buffer pressure: Flows beeinflussen sich nicht mehr so stark.

10. TCP Evolution

Selektive Quittungen
Ack like in RFC 793 have inefficient loss recovery:
only one lost Segment can be recovered p. RTT
multiple losses often lead to retransmission timeouts
and head-of-line blocking

NACK (negative ACK)
ACKs missing data explicitely
SACK (selective ACK)
explicitely ACK out-of-order data
on incoming „in-order-data": ACK as usual
„ „ out „ „ : SACK newly received segment
„ „ „ „ but adjacent to previously SACKed out-of order data: SACK whole block

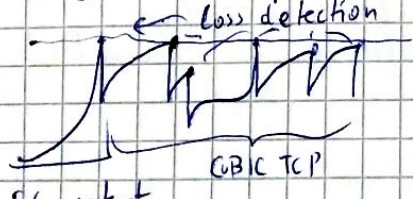
ACK=6 SACK=(8,10),(12,13)

Duplicate SACK (DSACK) Acks Segments that arrived multiple times
Forwarding ACK (FACK!) specifies when data may be sent during loss recovery - Replaces Fast Retransmit & Fast recovery

SYN-Cookies
against SYN Flooding ≙ spamming SYN-packets (TCP) w/ fake source address, Server can't handle more connections
Idea: code timestamp, MSS-size, cryptohash into seq number
decode ACK-1 of client and check for correct content
otherwise drop it ⇒ no connection
can't use special options that usally would be coded in seq number, MSS can't be arbitrary
⇒ use only in case of likely DOS-Attack

TCP @ high-speed

RTT = 100ms
Datarate D = 10 Gbit/s
MSS = 1500 B
using TCP Reno
assuming 1 packet drop each 5mrd. Segments
⇒ time needed to reach 10 Gbit/s (83k Segments/s) = 70min
idea: use multiple TCP connections asynchronily.
idea: increase congestion window faster.
TCP CUBIC
growth independent of RTT



Avg. datarates w/ AIMD

$D = \frac{1}{RTT} \cdot \sqrt{\frac{\alpha}{2}\frac{2-\beta}{\beta}\frac{1}{p}}$ $\alpha = 1, \beta = 1/2$

$\Rightarrow D = \frac{1}{RTT}\sqrt{3/2 \cdot \frac{1}{p}}$

to reach same increasion w/ CUBIC-TCP

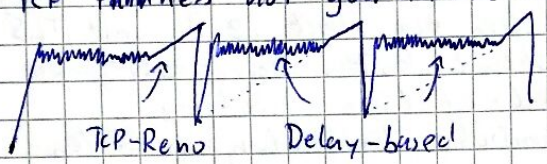
$W_{TCP} = W_{max} * (1-\beta) + \alpha \frac{1}{RTT}$

$\alpha = 3\frac{\beta}{2-\beta}$

Compound TCP (CTCP)

measures RTT growth ⇒ implicit Queeue growth
⇒ aggesive increase if queues are empty
$W = W_{RENO} + W_{delay}(RTT)$

shortest RTT measured = queues empty
needs to adjust RTT continously
measuring needs to be precise
TCP fairness not guaranteed if initial RTT was luckily low



TCP & Web

high delay in responses → less costumers on that site ⇒ less profit
⇒ aim to keep delay as low as possible
handshake, slow start, transmission of actual data ... world wide wait
≥ 2 RTT
⇒ minimum time for transmission: $2RTT + \frac{objectsize}{datarate}$
whereas 2RTT dominates for small objects
If $W = 2\,MSS$
K for transmitting object with size O: $K = \left\lceil \frac{O}{W * MSS} \right\rceil$

⇒ respond time $2RTT + \frac{O}{D} + (K+1)\left\lceil \frac{MSS}{D} + RTT - \frac{W \cdot MSS}{D} \right\rceil$
Server needs to wait for Ack of client ↑

If $\frac{W \cdot MSS}{D} \geq RTT + \frac{MSS}{D}$ ⇒ server can send w/o pauses

ELif $\frac{W \cdot MSS}{D} < RTT + \frac{MSS}{D}$ ⇒ server needs to wait for Ack

RTT dominates web-delay, because index.html often < 10 kB

Slow Start ⇒ small window size @ beginning ⇒ RTT dominates delay

Idea: increase initial MSS in Cwnd (today ~ 3-4)

TCP-Fast-Open (TFO)
Idea: Data request in SYN-Packet
⇒ −1 RTT
but opens room for attacks, DoS via SYN-Data Flooding
Source address spoofing
⇒ TFO Cookie
Client needs to request TFO Cookie via regular TCP
Server generates unique-only for this client working cookie
can be used afterwards
Server replies w/ Cookie in SYN-ACK Packet
Client can then use this cookie for later use
sends SYN + Cookie + Data request
Server validates source IP & Cookie and replies w/ SYN-ACK
and begins immediately w/ data sending
Server can disable cookie in case of DoS Attack assumption

Realitätscheck?

Früher war Endstelle im Protokollstack IP
ist heute ~~HTTP~~ HTTP
Anwendungen benötigen andere Kommunikationsabstraktionen (~~HTTP~~ HTTP!)

API
Klassisch: TCP-Bytestrom & Socket Interface
De-facto: HTTP-Interface
umfasst TCP als Teil des Substrates
Transportschicht ist heute eher ein Stack, z.B. mit TLS
Reihenfolgetreue bei TCP führt zu hohen Latenzen. Darüberliegende Anwendungen werden gebremst. Head-of-Line-Blocking
Streaming leidet (ohneeein :), dynamische Anwendungen leiden

SPDY und HTTP2
Sollen solche Probleme lösen
höhere Sicherheit
kurze Antwortzeiten
Rückwärtskompatibilität
nur Client und Server müssen angepaßt werden

Realisiert durch:
1 TCP Verbindung pro Server
⇒ keine erneuten 3-Wege-Handshakes
kein Slow Start
Multiplexing bei ~~Anfragen~~ ⇒ 1 Stream via TCP
Priorisierung " "
Server-Push ≙ senden an Client ohne dessen explizite Nachfrage
Effizientere Header
verpflichtende Nutzung von TLS

Framing
Liefert Einheiten für das Multiplexing
Data Frames, Fragmente der Daten
erfahrungsgemäß bessere Quality of experience
etwas mehr Overhead
Control Frames
um Stream und dessen Frames zu verwalten

Probleme
Anlaufschwierigkeiten
HTTP umging mit vielen Verbindungen dieses Problem
ist nun explizit verboten
höhere initCwnd
Slow-Start-after-idle sollte deaktiviert sein
Verlust eines Segments blockt alle gemultiplexten Streams
⇒ keine Lösung mit TCP
⇒ neues Protokoll

Quick UDP Internet Connections (QUIC)
0-RTT-Aufbau, ähnlich zu TFO
Reihenfolgetreue nur innerhalb eines Streams
TLS by default
Staukontrolle angelehnt an TCP
Fairness: eine QUIC mit n Streams verhält sich wie n TCP Verbindungen

QUIC übernimmt Aufgaben von HTTP2, TLS und TCP
optimiert: nutzt seq Num (zuverlässig) und Retransmissions
enhalten neue " "
Initialisierungsvektoren verschlüsselt

„0-RTT" durch gemeinsamen Handshake von Transport, Verschlüsselung und HTTP-Anfrage in der selben RTT
Nur möglich bei Bestand einer Verbindung vor Kurzem (oft bei Google der Fall)

QUIC setzt auf UDP auf.
Könnte direkt IP nehmen, ist aber mit NAT und Firewalls nicht gut kompatibel

MultiPath TCP (MPTCP)
für Geräte, die drahtlos sind und von Access-Point zu Access Point wechseln
Multi-Homing für Server-Farmen
Datenzentren mit redundanten Pfaden
Identifikation aber statisch über Quell-IP, Ziel IP, Quell-Port, Ziel-Port
keine Änderung während offener Verbindung möglich
⇒ TCP ist ein Single-Path-Protocol

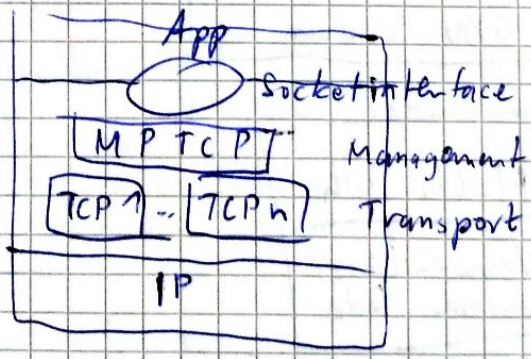
⇒ MPTCP
Beispiel: Iphone mit 3G & Wifi
Hat 2 verschiedene IP Adressen / Verbindung kann nicht migrieren

Ziel: Erweiterung für TCP zu Nutzung von mehreren Pfaden
+ Leistung
+ Availability

Muss Kompatibilität zu Anwendungen bereitstellen
sowie " zu allen TCP Geräten
Middle Boxes ändern z.T. TCP-Optionen und halten den Zustand

MPTCP-Verbindung
ist eine Kommunikationsbeziehung zwischen Sender & Empfänger
besitzt einen oder mehrere MPTCP-Subflows
MPTCP-Subflow
Beziehung zwischen Quell-IP-Adress und Ziel-IP-Adresse
≙ reguläre TCP-Verbindung
Reihenfolgetreu
Endet mit RST oder FIN
3-Wege-Handshake
können dynamisch zu einer Sitzung hinzugefügt oder gelöscht werden

Struktur



TCP Option MP-Capable im Handshake
Senden Token X, Y zum späteren Hinzufügen / Löschen eines Subflows mit

MP Join, Token

MPTCP Segmente haben zwei Seq nums. Eine für gesamte MP-Verbindung andere für den subflow
=> Lückenlose Nummern für Subflows

Beachten: Eingangspuffer: einer für die gesamte MPTCP Verbindung
Scheduling auf höherer Schicht, muss die Reihenfolge und Zuordnung erledigen

Staukontrolle wäre mit gleichem Algo unfair gegenüber TCP
neuer Ansatz:
bei Verlust, wie gehabt: Cwnd *= 0,5

Balance beim Anstieg: wenigen belastete Subflows dürfen stärker ansteigen
Congestion Avoidance:
~~für~~ Quittung, die im Subflow r empfangen wurde:

$$Cwnd_r = Cwnd_r + \min\left(\frac{\alpha}{Cwnd_{Gesamt}}, \frac{1}{Cwnd_r}\right)$$

Grundlegende TCP - Verfahren

TCP Tahoe
Slow Start Congestion Avoidance Fast Retransmit

TCP Reno
Wie Tahoe + Fast Recovery

Hohe Bandbreiten

Parallel
Mehrere parallele Verbindungen

CUBIC (Linux)
Kubische Funktion unabhängig von RTT

Compound (Windows)
Proaktiv RTT

Web-Basiert

Initiales Fenster
$C_{init} \geq 10$

Fast Open
Daten bereits mit SYN Senden

Data Center

Data Center
Verwendung von ECN

Multi Path
Verwendung mehrerer Pfade

11. Layer - 1

Binary → Kodierer → Medium → Dekodierer → Binary
↑ Informationsquelle — ↑ Störquelle — ↑ Informationssenke

Digitale Signale

Zeit diskret, t zwischen aufeinanderfolgenden -möglichen- Änderungen der Signalkoordinate
⇒ Schritt-Takt T

isochron ≙ zu clock, Zustandswechsel nur zum Takt
≠ anisochron

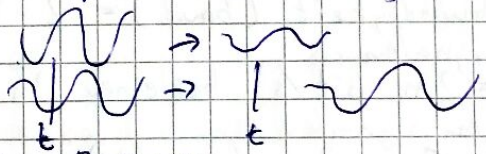
Baudrate ≙ Schrittgeschwindigkeit, # Signalwechsel / Zeiteinheit
baud ≙ (1/s)
bei isochronen Signalen 1/T
Auch als Baudrate bezeichnet
Übertragungsgeschwindigkeit : bit/s
Schrittgeschwindigkeit = Übertragungsgeschwindigkeit ?
nur, wenn Signal binär und wenn jeder Schritt als Signalelement genau ein bit als Codeelement darstellt

Mehrstufige Signale
Signalkoordinate kann mehr als 2 Werte annehmen
⇒ pro Zeit mehr bits

Störquellen

Analoge Signale haben kontinuierliches Frequenzspektrum
physikalische Medien haben nur ein endliches " zu Übertragung
⇒ Signale müssen darauf angepasst sein
mithilfe zweier Grenzfrequenzen: obere und untere
Bandbreite ≙ Breite des Spektrums, das übrig bleibt

Arten von Störungen
Dämpfungsverzerrung
Laufzeitverzerrung
Signalstörungen, wie:
- transient, stochastische Prozesse
- Weißes Rauschen
- Überlagerung von Nutzsignalen und Nebensprechen
- Echobildung
- Brumm Signale
- Störimpulse



Dämpfung — logarithmisch (Db)
Signalstärke nimmt mit steigender Distanz ab.
!! muss beim Empfänger noch deutlich größer sein als das Rauschen
Einsatz von Repeatern & Verstärkern
Dämpfung variiert mit der Frequenz
⇒ Einsatz von EQs

Verzögerungsverzerrung
bedingt durch Ausbreitungsverzögerung
auch Frequenzvariierend
⇒ Phasenverschiebung zwischen den Frequenzen
⇒ Intersymbol-Interferenz
beschränkt maximale Datenrate

Rauschen
thermisch, abhängig von der Temperatur
Gleichmäßig verteilt über die Frequenzen (weißes Rauschen)
kann man nichts gegen tun
Modulationsrauschen
durch Signalüberlagerung $f = f_1 + f_2$
Übersprechen
durch Kopplung zweier Kabel, die sich beeinflussen
Impulsrauschen
durch elektromagnetische Störungen, Blitze
analog: Knacken
Digital ... fatal
Unregelmäßiger Ausschlag mit hoher Amplitude

Kanalkapazität
≙ maximale Rate, mit der Daten über einen Kommunikationskanal gesendet werden können

Datenrate ≙ bit/s zur Kommunikation
Bandbreite ≙ höchste - niedrigste Frequenz [Hz]
Rauschen
Fehlerrate = Error/s 0 statt 1 oder 1 statt 0

Ziel: Maximiere Datenrate bei gegebener Bandbreit und minimiere dabei die Fehlerrate

Nach Nyquist gilt:
max. Schrittgeschwindigkeit [baud] = 2 · Bandbreite [Hz]
bei binären Signalen:
Datenrate = 2n bit/s bei einer Bandbreite von n Hz
bei mehrwertigen Signalen:
Datenrate $C = 2n \log_2(M)$
M ≙ Signalwerte
Erhöhung von M führt zu einer Erhöhung der Datenrate

bei hohen Datenraten sind mehr bits von einer Störung betroffen, als bei gleicher Störung und niedrigerer Datenrate
Shannon-Hartley-Gesetz
Signal to noise ratio (SNR)

$$C = B \log_2(1 + SNR)$$

C ≙ Datenrate [bit/s]
B ≙ Bandbreite [Hz]
S ≙ Energie des Signals
R ≙ " der Störung

$$SNR\ [dB] = 10 \log_{10} S/R$$

Digitale Leitungscodes

Kodierung beim Sender

Sender → Quellencodierer → Kanalkodierer → Leitungskodierer → Modulator
0101111001 → Komprimiert → Erhöht Robustheit CRC... → Vorbereiten → Binden an phys. Signale

Taktrückgewinnung

erforderlich, falls keine separate Taktleitung dabei

Gleichstromanteil

Wegen der angeschlossenen Geräte möglichst vermeiden
Bewertung RDS (Running digital sum)
Gleichstromfrei, wenn statistisch bei 0

Baudrate

hoch → viele Wechsel → hohe Anfälligkeit → geringere Reichweite

Komplexität

Leitungscodes

- Binär — Symbolwert durch Signalwert
  NRZ
- Biphase — Symbolwert durch Phasensprung
  NRZI, Manchester
- Ternäre — Symbolwert 0,1 → Signalwert 0,1,-1
  RZ, AMI
- Blockcodes — m Bits werden zusammengefasst und auf ein n-langes Wort abgebildet
  N-äre Darstellung
  4B/5B ... 64/66B

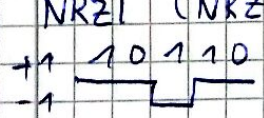
Beispiele

NRZ (non-return to zero)

1 0 1 1 0

- Taktrückgewinnung
- RDS → +∞
- + Baudrate

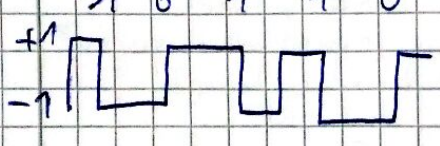
NRZI (NRZ inverted)



Wechsel bei "1"
- Taktrückgewinnung
- RDS → +∞
+ Baudrate

Manchester

1 0 1 1 0



"1" Fallende Flanke in der Mitte des Intervalls
"0" Steigende " " " " "
Hilfswechsel einfügen bei mehrfachem gleichen Symbol

+ Taktrückgewinnung
+ Komplexität (EQ-Gatter zwischen Daten & Takt)
+ RDS
- Baudrate

Effizienz im worst-case: 50%

RTZ (Return to zero)

1 0 1 1 0

\+ Taktrückgewinnung
− Baudrate
− Gleichstromanteil (RDS)

AMI

1 0 1 1 0

"0" = 0
"1" = abwechselnd −1 oder 1

− Taktrückgewinnung
\+ RDS
\+ Baudrate

Modifizierte AMI Codes

Verstoße gegen die Regel der abwechselnden pegel bei einer 1, um Taktrückgewinnung zu gewährleisten
0-Folge gewisser Länge wird durch Füllsequenz gleicher Länge ersetzt
Codeverletzungen erhöhen Taktgehalt
Erhaltung der RDS
Empfänger muss das erkennen und rückgängig machen

⇒ B8ZS

8×"0" → 000+−0−+ bei letzter 1 positiv
→ 000−+0+− " " " negativ

1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1

HDB3

4×"0" → "1" seit letzter Ersetzung

letzte 1	gerade	ungerade
negativ	000−	+00+
positiv	000+	−00−

+RDS ☺

Block codes

Verbessern der RDS, Taktrückgewinnung von NRZ-Codes

Block$_1$ / Block$_2$ - Notation

Block$_1$ ≙ Daten
Block$_2$ ≙ Codewort

Block$_2$ wird so gestaltet, dass er Redundanz ermöglicht
hat mehr Wörter, als Block$_1$ (mehr Bits!
darf höherwertig als Block$_1$ sein
nutzt nur eine Untermenge seiner möglichen Zustände
Ausnutzung von Redundanz für
Taktrückgewinnung und Gleichstromfreiheit

4B/5B Code

Block$_1$: Länge 4 bit, binär
Block$_2$: länge 5bit, binär

Auswahl von Block$_2$: nicht mehr als eine führende 0
" " " zwei abschließende 0en

Effizient : 80% = 4/5

↳ 1000 0000 0000 0001 1111
↳ 10010 11110 11110 01001 11101

Verbleibende Codewörter sind entweder ungültig oder werden anders interpretiert, zum Beispiel als Start-of-Frame-Delimiter

8B/10B

4B/3T
Auswahl von Block$_2$ mit dem niedrigsten Gleichstromanteil
Keine Verwendung von 000

64B/66B
Overhead ~3%
Voranstellen von 2 bit Synchronisation

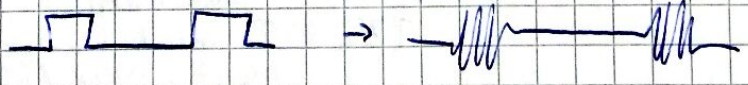
Modulationsverfahren

digitales Signal → analoges Signal
Grundlage Fourier-Analyse

$$x(t) = \frac{C_0}{2} + \sum_{n=1}^{\infty} C_n \cos(2\pi n f_0 t + \theta_n)$$

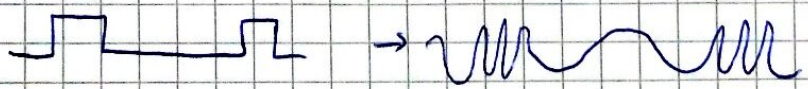
C_n: Amplitude, f_0: Frequenz, θ_n: Phase

Amplitudenumtastung
Signal (binär) schaltet zwischen Oszillator und GND



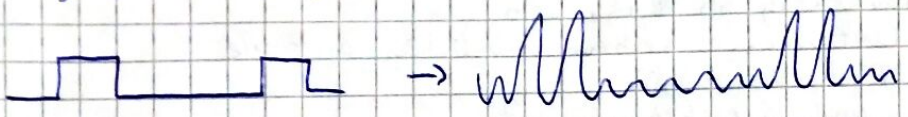
technisch einfach
wenig Bandbreite
störungsanfällig

Frequenzumtastung
Signal schaltet zwischen zwei Oszillatoren um



größere Bandbreite
mit oder ohne Phasen-
sprünge

Phasenumtastung
Signal hält Pegel bei 0, wechselt bei 1

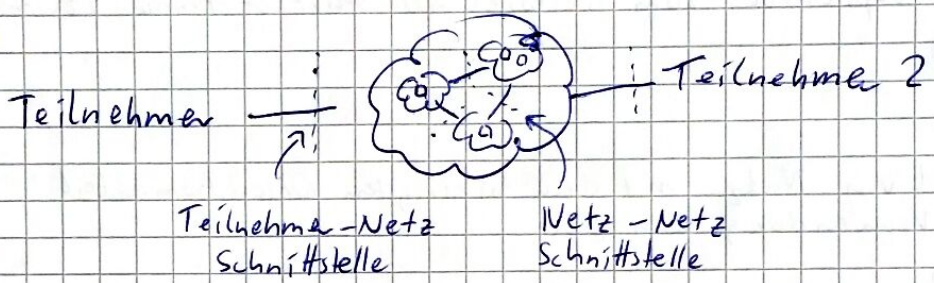


komplexe Demodulation mit Taktrückgewinnung
relativ störungssicher

12. ISDN (Integrated Services Digital Network)

Leitung ≙ Reservierung von Ressourcen für die Übertragung von Nutzdaten
keine physikalische Leitung
Ressourcen werden fest einer Verbindung zugeteilt
Keine Metadaten / Pufferüberläufe
Aber ggf. schlechte Ressourcennutzung
Leitungsaufbau durch Signalisierung
Routing und Zuweisung / Rücknahme von Ressourcen
i.d.R. Einsatz von Paketvermittlung

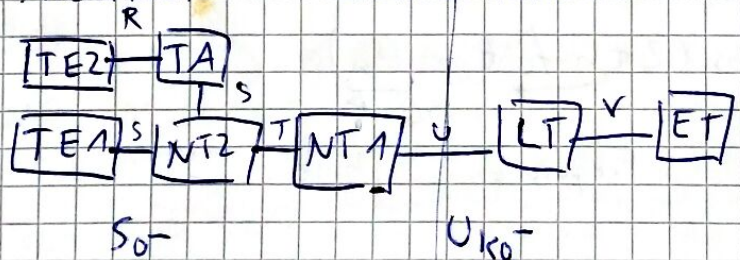
Aufbau



Hierarchie intern
Zentral (vollvermascht)
Hauptvermittlung
Knotenvermittlung
Ortsvermittlung

Ziel von ISDN
Digital bis zum Teilnehmer
Integration unterschiedlicher Dienste (Bild, Ton, Sprache, Daten)

Architektur Teilnehmer Netzseite



ET ≙ Exchange Termination
Vermittlungsabschluss
LT ≙ Line Termination
NT1 ≙ Network Termination 1
Netzabschluss Schicht 1
NT2 ≙ Network Termination 2
Netzabschluss Schicht 1-3
TE1 ≙ Terminal Equipment 1
ISDN-Fähig
TE2 ≙ Terminal Equipment 2
nicht-ISDN fähig
TA ≙ Terminal Adaptor
für TE2

Aufgabe des NT
Übertragungstechnischer Abschluss
der Netzseite (U_{K0} - Schnittstelle)
und der Teilnehmerseite (S_0 - Schnittstelle)
Speisung der Teilnehmerinstallation
220V - Netz
Erkennung von Rahmenfehlern

NT1:
Taktrückgewinnung
Rahmensynchronisation
Echo-Kanal-Steuerung
NT2:
Vermittlungsfunktionen
Ist optional

ET: (Schicht 1-3)
Medienzugriff auf D-Kanal (Schicht 2)
Signalisierungsfunktionen in Schicht 3
Multiplex- und Demultiplexfunktionen
LT (Schicht 1)
Umsetzen der Übertragungsverfahren
Ableiten und Generieren von Takten
Fernstromversorgung des Teilnehmerbereiches

Basis-Anschluss

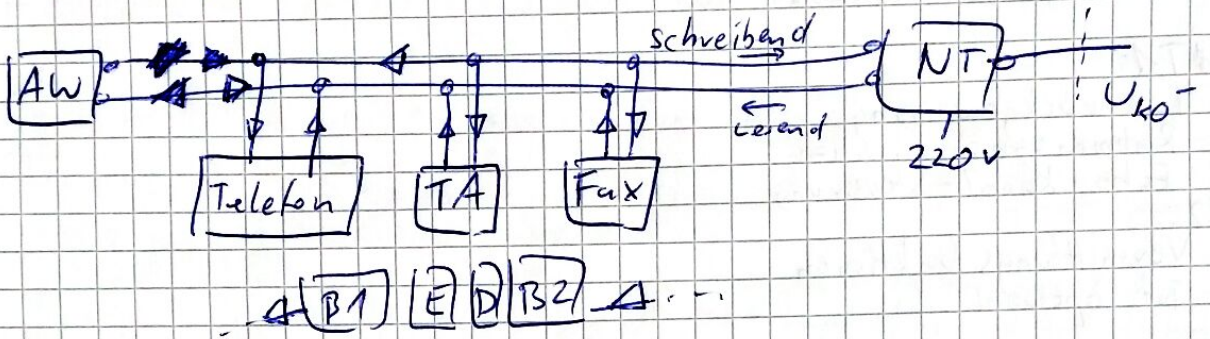
2 B + 1 D Leitungen (2 · 64 kbit/s + 16 kbit/s)

B-Kanal (Schicht 1-7)
zur Nutzdatenübertragung
operiert unabhängig von anderen D-Kanälen
können verschiedene Zielrichtungen haben
" " Datentypen übertragen
" gleichzeitig operieren
Medienzuteilung fest
wird dediziert zwei Endpunkten zugeteilt
diese Zuteilung ist Aufgabe des D-Kanals

D-Kanal (Schicht 1-3)
16 kbit/s schnell (oder langsam kol)
Bidirektional zwischen NT und Teilnehmer
Koordiniert bis zu 8 Teilnehmer
CSMA/CD Variante

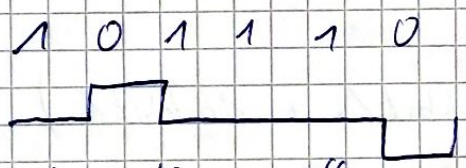
E(cho)-Kanal
16 kbit/s
Richtung NT → Endsystem
Für Medienzugriff erforderlich
Carrier Sensing (CS)
Collision Detection (CD)

Teilnehmer Schnittstelle S_0
Doppelader in jede Richtung, Simplex
Verschiedene Topologien möglich
Raummultiplex – Richtungstrennung
Zeitmultiplex für andere Geräte in der Topologie



Echo-Kanal
Aufgabe: Unterstützung des konkurrierenden Medienzugriffs
Problem: Endsystem kann auf ausgehenden D-Kanal nicht lesen
Lösung: NT reflektiert D-Kanal in E-Kanal

Leitungskodierung: Inverser AMI-Code
1 ≙ Nullpegel
0 ≙ Wechselnd 1 oder -1



D-Kanal-Medienzugriff
falls 8 bit$_{lang}$ nichts auf dem E-Kanal liest → frei

Kollisionserkennung durch sendende Station
Senden ist 1-persistent
Anderes Signal auf E-Kanal als D-Kanal?
0 überschreibt 1) Aufhören zu senden, weiter prüfen
kein exponentieller Backoff
Andere sendende Station merkt davon nichts
Wieso 8 bit warten?
Inverser AMI, kann keine 8 bit lang keine Aktivität haben → frei
HDLC Protokoll auf Schicht-2
Frame begrenzung durch Flag 01111110
Bitstopfen erhält Datentransparenz
bei 5 "1" wird eine "0" gestopft
Empfänger muss diese wieder entfernen
⇒ Länge des Pakets auch abhängig vom Inhalt
Bei gleichzeitigem Beginn, ist für beide die Teilnehmerkennung im E-Kanal falsch.

D-Kanal & Signalisierung
Q.931 Protokoll steuert SS-7
⇒ Aufbau / Abbau von Verbindungen zwischen Endsystemen
Und Art des Dienstes: Rückruf / Datenanfrage...
Benötigt ISDN eindeutige Adressen für Schicht 3

ISDN Adressen

Mögliche → Ausscheide-Ziffern | Landeskennzahl | Ortskennzahl | Teilnehmernummer | Subadresse

ISDN Nummer (max 15 Ziffern) | max 32

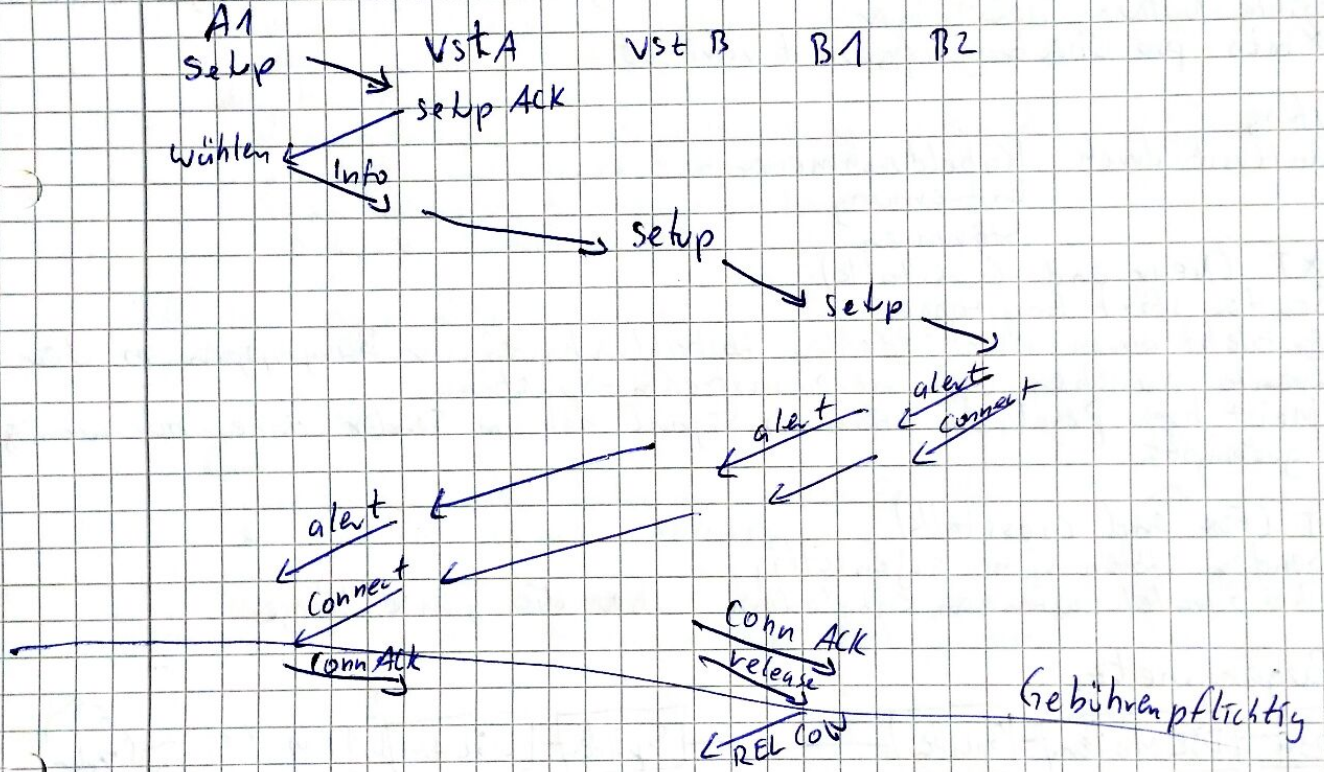
ISDN Adresse

für Netzübergänge

0,00... bei nationalem Netz

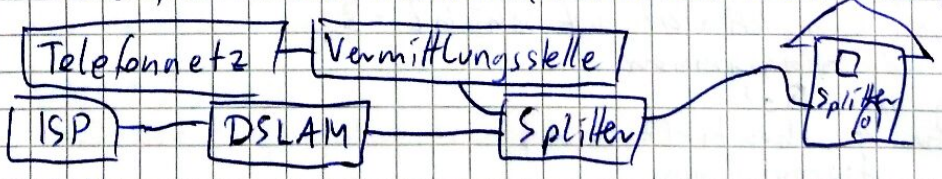
1,11... bei internationalem Netz

Adressierung einer T-Schnittstelle bei ISDN Rufnummer
S- " " " Adresse



DSL (Digital Subscriber Line)

Ziel: Leistungsfähige Leitung für Teilnehmer
asymmetrischer Anschluß: 8 Mbit/s Download => ADSL
576 kbit/s Upload
Bedingt durch das Modell: kleine Anfrage, große Antwort
Symmetric DSL (SDSL)
oft für Geschäftskunden, reiner Datenanschluss, keine Telefonie



Splitter
Trennt Signal in Telefon / Datensignal
Sowohl bei der Vermittlungsstelle als auch beim Kunden vorhanden
Telefonie ist höher prior als Daten - Bei Ausfall bleibt diese erhalten

DSLAM (DSL Access Multiplexer)
Gegenstück zum DSL-Modem beim Teilnehmer

Datenübertragung

Telefonie / Daten in Frequenz geteilt

3 kHz für analoge Telefonie
0–138 kHz für digitale "
138–276 kHz für DSL Upstream
276–1104 kHz " " Downstream

DMT (Discrete Multitone Modulation)
Telefonie: Träger 1–31
DSL Uplink: " 32–64
DSL Downlink: " 65–256

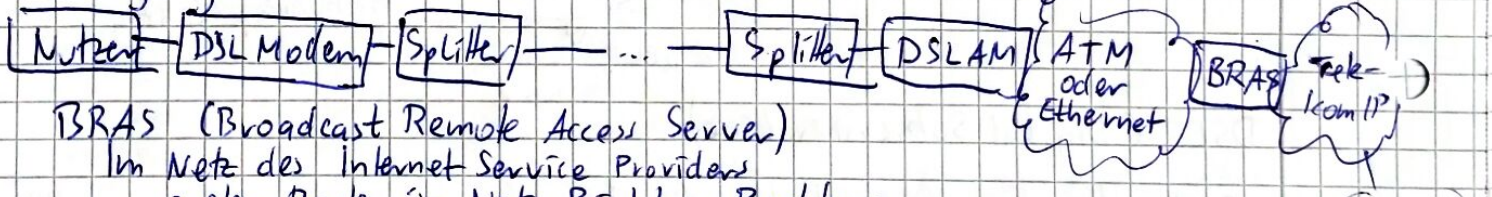
parallele Übertragung auf mehreren orthogonalen Trägern (niedrige Datenrate)
Flexible Nutzung der Träger
bits pro Übertragungsschritt variiert

Dämpfung
beeinflusst durch Kabeldurchmesser
Entfernung
Störungen

NEXT (Near end Crosstalk)
Sender stört Empfänger
Entsteht durch nicht ideales Abstrahlverhalten der Baugruppen an der Grenze zwischen Up- und Downstream-Spektrum
Meist hoher Pegel, da störendes Signal nah am Sender, daher nur wenig gedämpft

FEXT (Far end Crosstalk)
Sender stören sich gegenseitig
Bei parallel laufenden Kabeladern über die ganze Länge

DSL-Zugangsnetz



BRAS (Broadcast Remote Access Server)
Im Netz des Internet Service Providers
... erster Router im Netz Richtung Backbone
Aufgaben
Aggregierung mehrerer DSLAM-Verkehre
Bereitstellung von Konnektivität auf Schicht-2
PPP-Sitzung (über Ethernet oder ATM (Asynchronous Transfer Mode))
IP (über ATM)
Bereitstellung von Konnektivität auf Schicht-3
Routing von IP-Datagrammen
Berücksichtigung von QoS
Schnittstelle zu Authentifizierung
Autorisierung
Accounting
RADIUS Protocol (Remote Authentification dial in user Service)
auf Schicht-7

Aufbau einer DSL-Verbindung

Erfolgt über PPP in 3 Phasen

1. Establish → LCP
   - Aufbau einer PPP-Verbindung
   - Aushandlung der Verbindungsparameter (Datenrate / Träger)
   - " " Authentifizierungsmethode
2. Authentifikationsphase
   - basierend auf der ausgehandelten Methode (PaP, Challenge Handshake)
3. Network-Phase
   - Vergabe der IP-Adresse
     - über IPCP, das PPP Network Control Protocol (NCP) für IP
   - Bekanntgabe der Adresse des DNS-Servers

Aushandlung der Datenrate

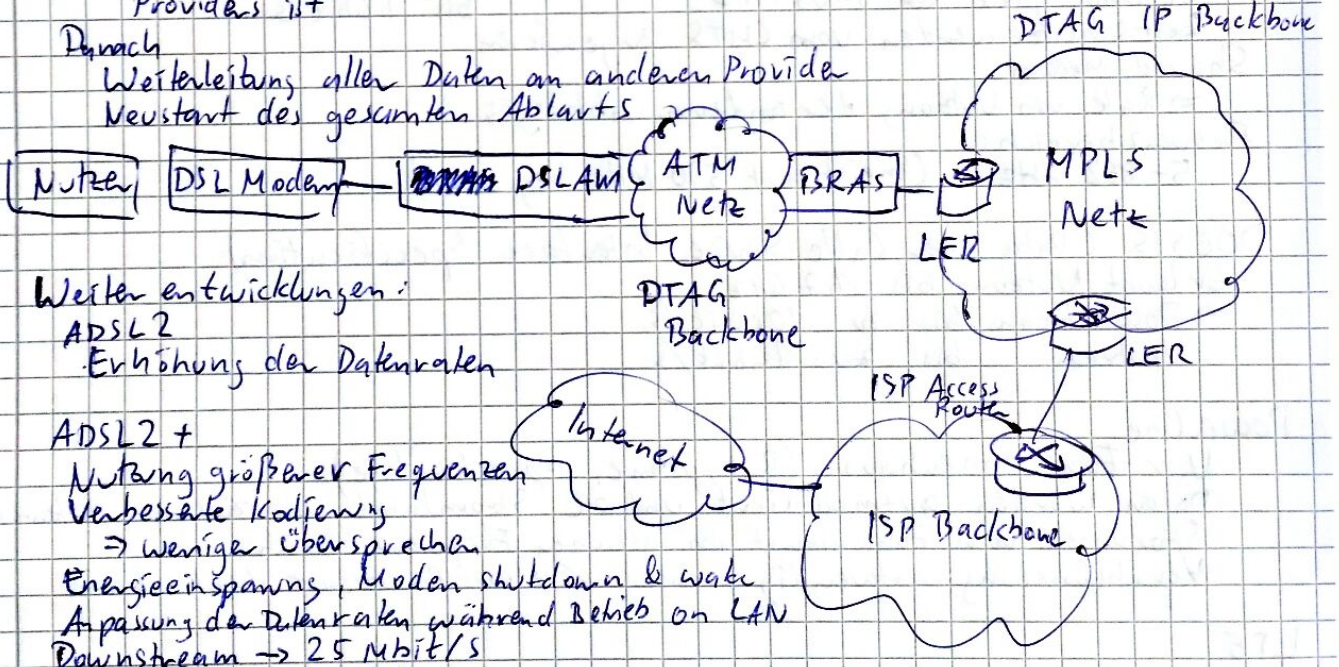
Fixed Rate
- Bestimmter Wert, Information aus Dämpfungstabelle
- enthält Sicherheitsaufschlag zum Erreichen der Datenrate
- auch bei einer Störung eines neuen DSL Anschlusses

Adaptive Rate
- Aushandlung der aktuell maximalen Datenrate
- oft höher als Fixed Rate
- Resynchronisation bei Verschlechterung notwendig
  - ⇒ kurzer Ausfall

Aufbau einer ADSL-Verbindung

- Abbruch des vorherigen Ablaufs in Authentifizierungsphase
  - Erst zu diesem Zeitpunkt ist klar, dass der Nutzer Kunde eines anderen Providers ist

Danach
- Weiterleitung aller Daten an anderen Provider
- Neustart des gesamten Ablaufs



Weiterentwicklungen:

ADSL2
- Erhöhung der Datenraten

ADSL2+
- Nutzung größerer Frequenzen
- Verbesserte Kodierung
  - ⇒ weniger Übersprechen
- Energieeinsparung, Modem shutdown & wake
- Anpassung der Datenraten während Betrieb an LAN
- Downstream → 25 Mbit/s
- Upstream → 1 Mbit/s

VDSL1 / VDSL2
- Größerer Frequenzbereich ⇒ höhere Datenraten ⇒ weniger Strecke möglich

Outdoor DSLAM
- Glasfaser bis zum Kasten ~ 500 m von Nutzer entfernt
  - ⇒ VDSL2 möglich - 50Mbit/s kein Problem (bis 200Mbit/s)

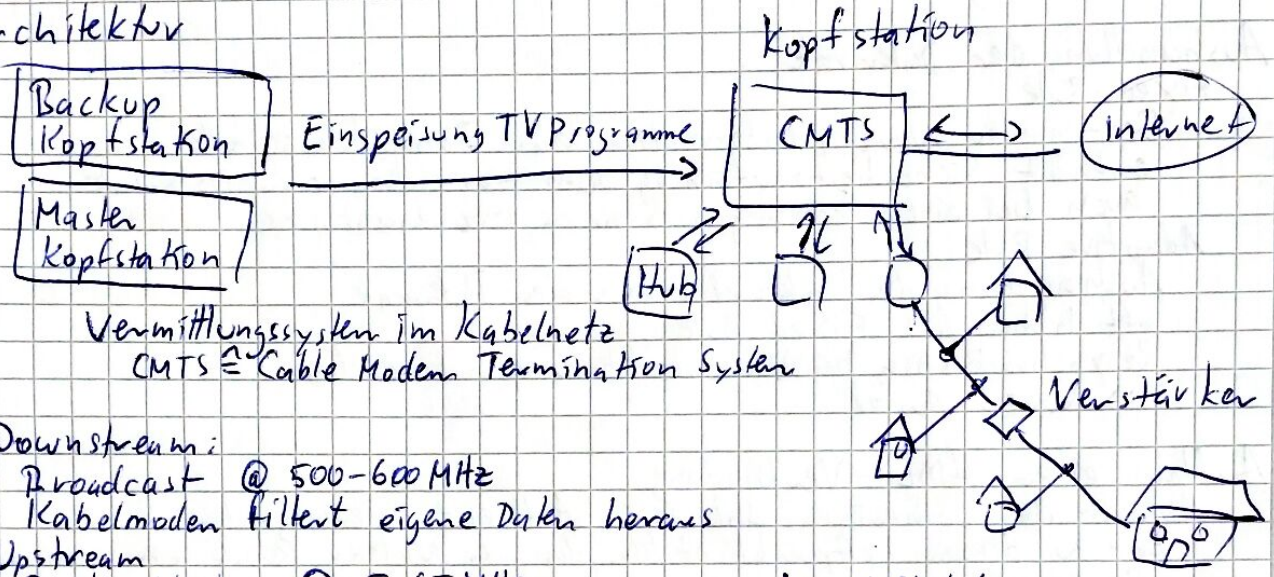
TV-Kabelnetz

Aufgebaut in den 80ern
zuerst ohne Rückleitung
erneuter Ausbau
heute auch für Telefonie & Internet

Topologie
Baum nur auf letzter Meile
Ansonsten stark vermascht

Erreichbarkeit
DE: ~75%

Architektur



Vermittlungssystem im Kabelnetz
CMTS ≙ Cable Modem Termination System

Downstream:
Broadcast @ 500-600 MHz
Kabelmodem filtert eigene Daten heraus

Upstream
Zeitmultiplex @ 5-65 MHz bis 1 Gbit/s
Zeitschlitze werden vom CMTS zugewiesen

Shared Medium
⇒ Rate von Nutzung der anderen abhängig

Frequenzbereich:
5-862 MHz (meist auf 570 MHz begrenzt)

DOCSIS (Data over Cable Service Interface Specification)
Erlaubt Nutzung bis 1,7 GHz
⇒ Downstream bis zu 10 Gbit/s
Upstream bis zu 1 Gbit/s

Powerline
Nur für Innenhaus (Babyphone, Sprechanlage)
Daten werden aufmoduliert und die Stromkabel wirken wie Antennen
Störungen vom Funkdienst im gleichen Frequenzbereich
Verschlüsselung notwendig, sonst wireless-Broadcast durchs Land

LTE
Für Flaches Land
wie DSL nur drahtlos
geteiltes Medium