

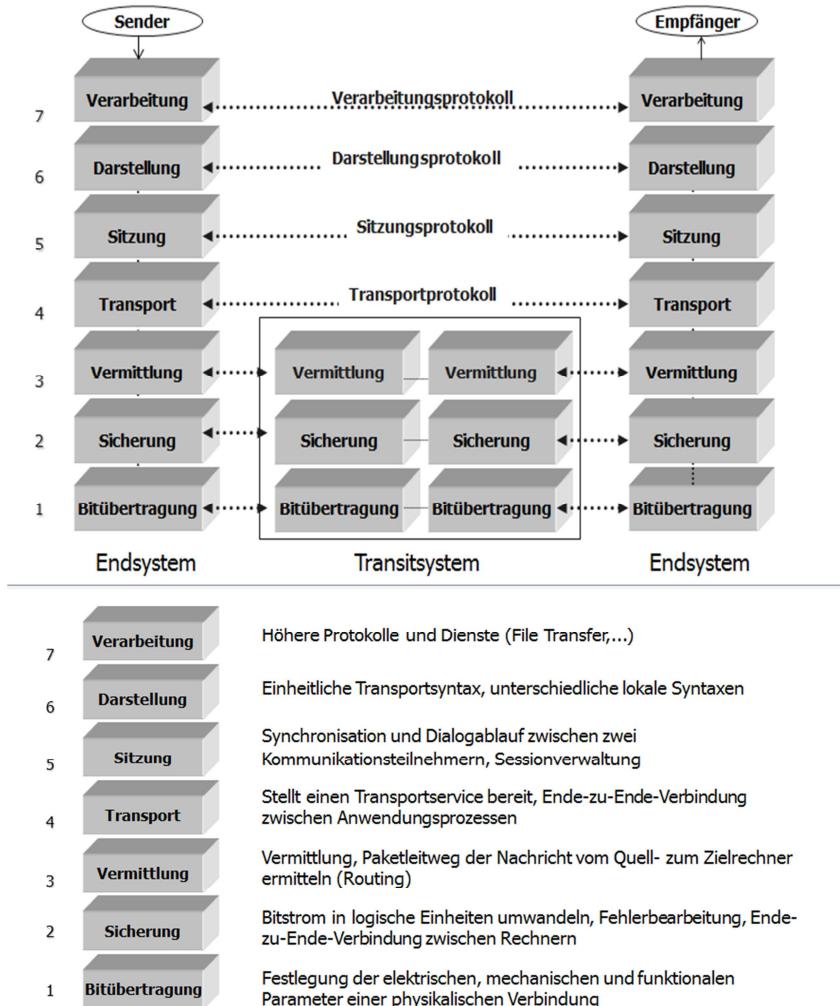
Zusammenfassung DaKo

Technische Grundlagen der Datenkommunikation

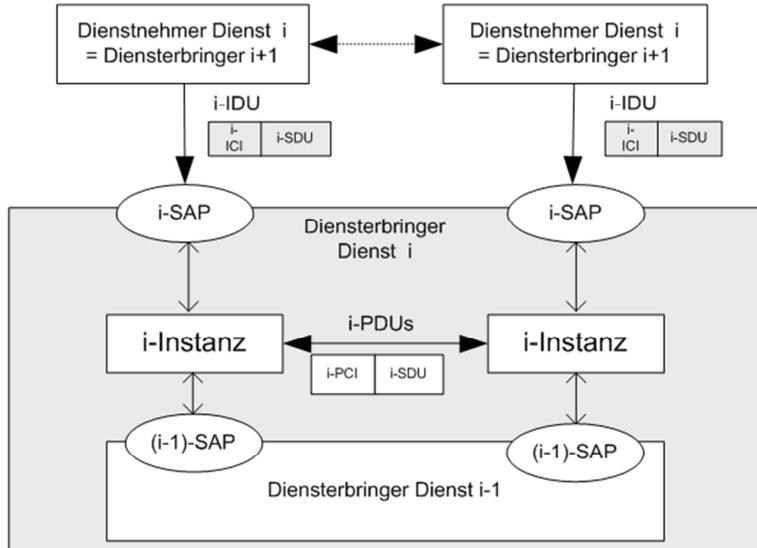
- Kommunikation zw. Rechnern in offenen, heterogenen Systemen wird beschrieben durch Referenzmodelle
- Vorteil: Offene allgemein verbindliche Vorstellung eines Kommunikationsvorgangs in Form eines Architekturmodells
- Beispiele:
 - ISO/OSI-Referenzmodell
 - OSI = Open System Interconnection
 - ISO = International Standardization Organization
 - TCP/IP-Referenzmodell

Die gesamte Funktionalität, die hinter der Rechnerkommunikation steckt, ist zu komplex, um ohne weitere Strukturierung verständlich zu sein. Im Rahmen der Standardisierungsbemühungen der ISO hat man sich daher gemäß dem Konzept der virtuellen Maschinen mehrere Schichten ausgedacht, um die Materie etwas übersichtlicher zu beschreiben.

- Um ein Kommunikationsprotokoll überschaubar zu machen, zerlegt man es in **Schichten** (layer)
 - Beim ISO/OSI-Modell sind es 7 Schichten
- Das **Endsystem** umfasst alle sieben Schichten
- Das **Transitsystem** umfasst nur die unteren Schichten, z.B. die Schichten 1–3

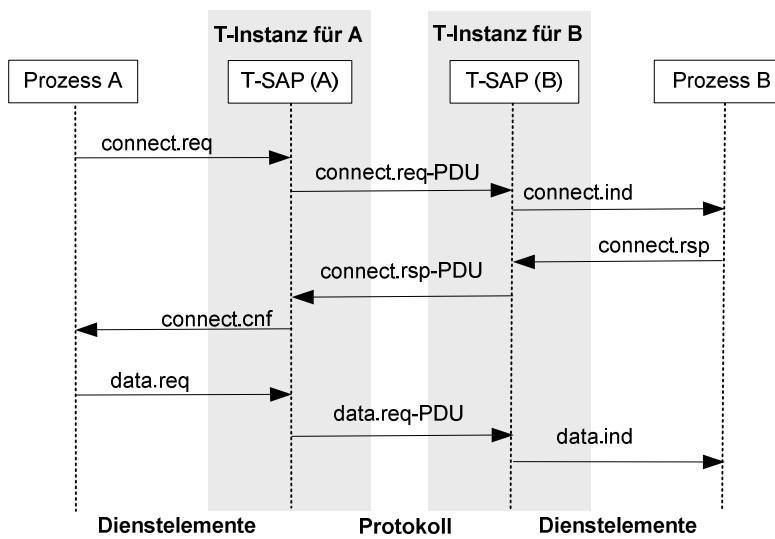


- Protokolle sind **Verhaltensrichtlinien**, auf deren Grundlage Daten zwischen Computern bzw. Prozesse untereinander unterhalten/verstehen/austauschen.
 - bestimmter Spielregeln, an die sich Sende- und Empfangsstationen halten müssen,
 - damit die Übertragungswünsche der Netzteilnehmer nicht im Chaos enden
- Protokolle sind **Vorschriften und Konventionen** zur Regelung von
 - Verbindungsaufbau
 - Nachrichtenübermittlung und
 - Verbindungsabbau
- Jede Schicht bietet der ihr jeweils übergeordneten Schicht Funktionen, sog. **Dienste** an
- Jede Schicht (bis auf die unterste) kann von der direkt darunter liegenden Schicht Dienste in Anspruch nehmen, **ohne ihre Implementierung zu kennen**
- Protokolle übernehmen verschiedene Aufgaben (je nach Schicht)
 - Verbindung aufbauen und Verbindung abbauen
 - Datenübertragung
 - Fehlererkennung und Fehlerbehebung
 - Staukontrolle (Congestion Control)
 - Flusskontrolle
 - ...

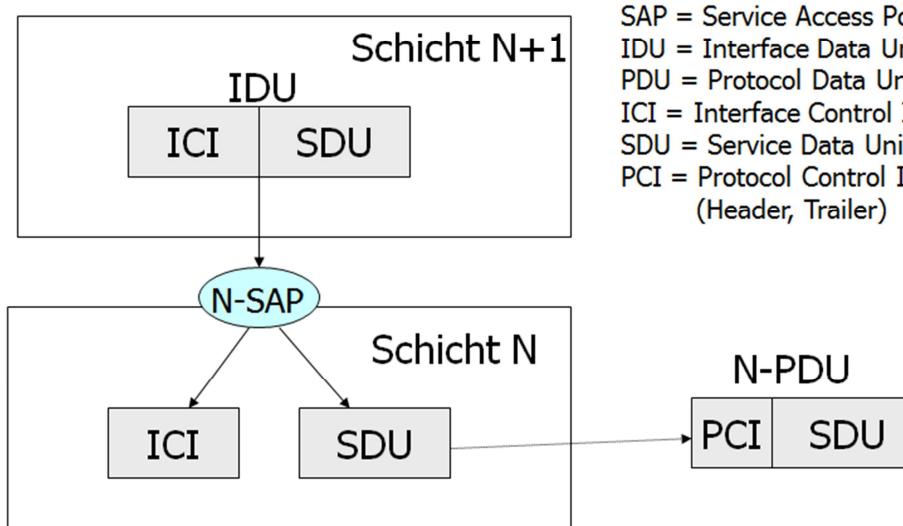


- Der **Dienstnehmer** in einer Schicht i nutzt zur Kommunikation einen **Diensterbringer** i (Dienstprovider) der wiederum die darunter liegende Schicht (i-1) nutzt
- Die Dienste werden über **Dienstzugangspunkte** (Service Access Points, SAP) bereitgestellt
 - SAPs sind logische Schnittstellen, deren konkrete Realisierung z. B. in Form einer Funktionsbibliothek oder als eigener Systemprozess gegeben sein kann
 - Die Funktionen innerhalb einer Schicht werden von einer **Instanz** ausgeführt
- Eine Instanz erbringt die Dienstleistung, die ein Dienstnehmer von einem Dienstprovider erwartet
 - Instanzen sind auf den kommunizierenden Systemen verteilt
 - Instanzen der gleichen Schicht kommunizieren miteinander über Protokolle
 - Zwei kommunizierende Instanzen werden als Partnerinstanzen bezeichnet
- Dienste (Abstraktion) im ISO/OSI-Modell
 - Beispiel: connect, disconnect, data (Datenübertragung)
- Dienstelemente/Dienstprimitive sind Operationen eines Dienstes
- Typische Dienstprimitive sind
 - Request
 - Indication
 - Confirmation
 - Response
 - Beispiel: connect.req, connect.ind, data.req

Zusammenspiel von Diensten und Protokollen



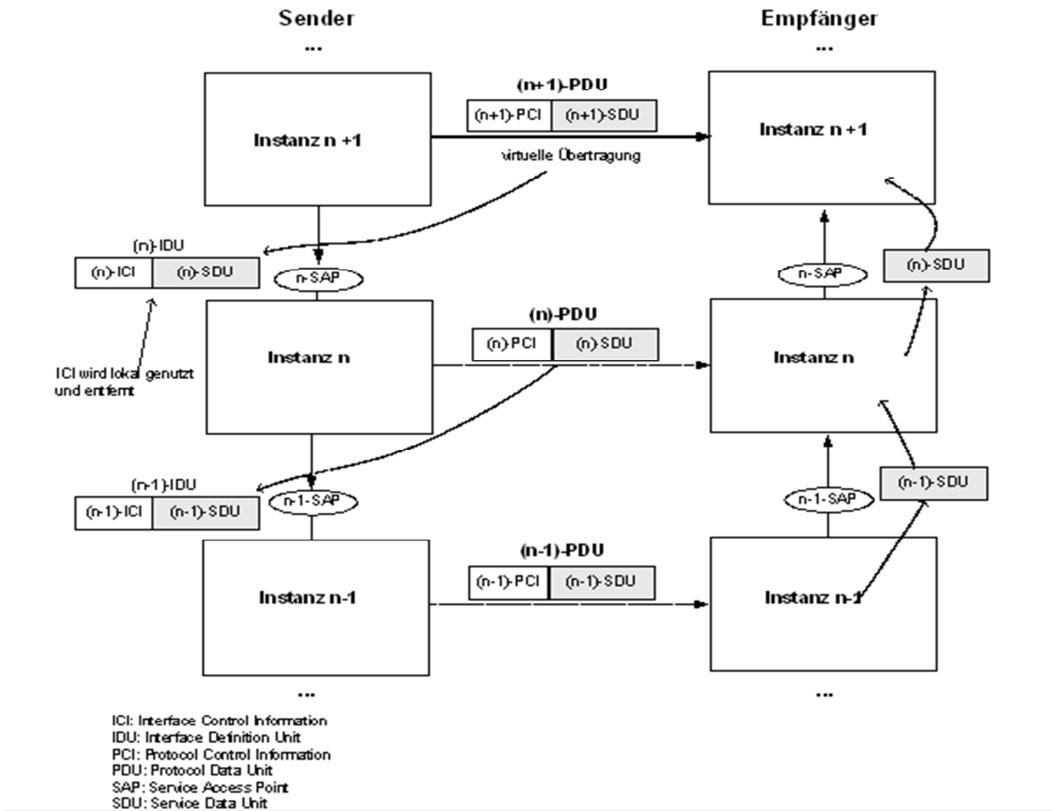
Allg. Begriffe der einzelnen Schichten und deren Zusammenspiel untereinander



SAP = Service Access Point
 IDU = Interface Data Unit
 PDU = Protocol Data Unit
 ICI = Interface Control Information
 SDU = Service Data Unit
 PCI = Protocol Control Information
 (Header, Trailer)

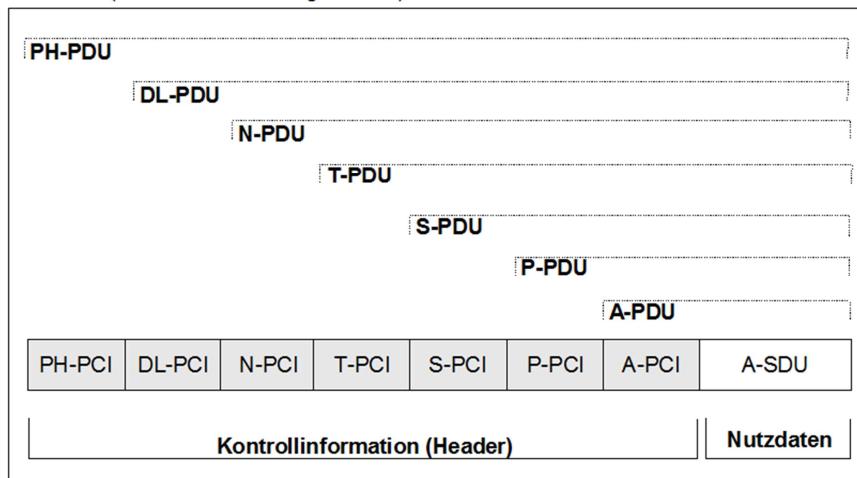
Weitere Begriffe aus dem Modell sind PDU, IDU, ICI, SDU, PCI und IDU, die hier noch kurz erläutert werden:
Nachrichteneinheiten, also Datenpakete die übertragen werden, bezeichnet man als Protocol Data Units (PDU).

- Unter einer IDU (Interface Data Unit) wird eine Schnittstelleneinheit verstanden, die eine Schicht n+1 einer Schicht n übergibt. Diese Nachricht besteht aus einer ICI (Interface Control Information) und einer SDU (Service Data Unit). Die ICI stellt die Steuerinformation für die Schicht n bereit und wird in der Schicht n interpretiert. Die SDU ist die eigentliche Nutzinformation, die übertragen werden soll.
- Die SDU wird von der Instanz in eine PDU eingepackt und mit einem Header, der sog. PCI (Protocol Control Information) versehen. Die Instanz der Sendeseite sendet die gesamte PDU logisch über das Netz zum Empfänger. Dort wird die PCI von der korrespondierenden Empfänger-Instanz interpretiert und die SDU wird nach oben zur Schicht n+1 weitergereicht. Die eigentliche physikalische Übertragung findet natürlich in Schicht 1 statt und aufgrund der Schichtenbearbeitung bildet sich eine Gesamt-PDU mit sieben PCIs, die tatsächlich übers Netz geht.



Der Aufbau einer PDU (aller 7 Schichten)

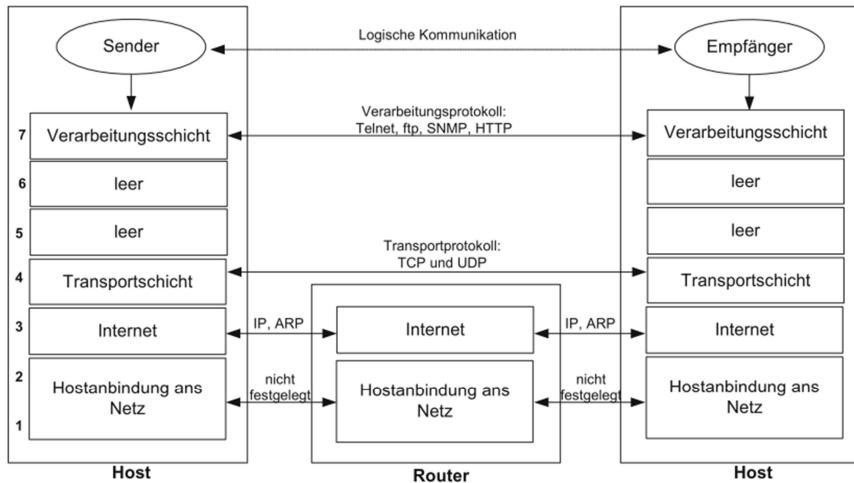
A-PDU (PDU der Anwendungsschicht)



Schichtenbezogene Namensgebung nach OSI

Services	SAP	Instanz	SDU	PCI	PDU
Physical	PH-SAP	PH-Instanz	PH-SDU	PH-PCI	PH-PDU
DataLink	DL-SAP	DL-Instanz	DL-SDU	DL-PCI	DL-PDU
Network	N-SAP	N-Instanz	N-SDU	N-PCI	N-PDU
Transport	T-SAP	T-Instanz	T-SDU	T-PCI	T-PDU
Session	S-SAP	S-Instanz	S-SDU	S-PCI	S-PDU
Presentation	P-SAP	P-Instanz	P-SDU	P-PCI	P-PDU
Application	A-SAP	A-Instanz	A-SDU	A-PCI	A-PDU

Das von der Internet-Gemeinde entwickelte TCP/IP-Referenzmodell ist heute der Defacto-Standard in der Rechnerkommunikation. Es hat vier Schichten, wobei die Schicht 3 (Internet-Schicht) und die Schicht 4 die tragenden Schichten sind. In der Schicht 3 wird neben einigen Steuerungs- und Adressierungsprotokollen (ARP, ICMP) 6 im Wesentlichen das Protokoll IP (Internet Protocol) benutzt. In Schicht 4 gibt es zwei Standardprotokolle. Das mächtigere, verbindungsorientierte TCP (Transmission Control Protocol) und das leichtgewichtige, verbindungslose UDP (User Datagram Protocol). TCP gab dem Referenzmodell seinen Namen.



Im Gegensatz zum OSI-Modell wird im TCP/IP-Modell nicht so streng zwischen Protokollen und Diensten unterschieden. Die Schichten 5 und 6 sind im Gegensatz zum OSI-Modell leer. Diese Funktionalität ist der Anwendungsschicht vorbehalten, d.h. die Verwaltung evtl. erforderlicher Sessions und die Darstellung der Nachrichten in einem für alle Beteiligten verständlichen Format muss die Anwendung selbst lösen. Kritiker des OSI-Modells sind der Meinung, dass gerade diese Funktionalität dieser beiden Schichten sehr protokollspezifisch ist.

Das war es zu den Grundlagen....

Bitübertragungsschicht

Aufgaben der Bitübertragungsschicht

- Bereitstellung der physikalischen Verbindung, Zugang zum Medium
- Festlegung der elektrischen, mechanischen und funktionalen Parameter
- Physikalische Bitdarstellung für das benutzte Übertragungsmedium
- Übertragung von Bits und Bitgruppen

- Die Zeit T, die zur Übertragung von Informationen benötigt wird, hängt von
 - der Kodiermethode (befasst sich mit der Abbildung eines bei der Quelle gegebenen **Symbolvorrats** auf den hierfür auszugebenden Signalverlauf im Kanal)
 - und
 - der Schrittgeschwindigkeit ab (Die Anzahl der Zustandsänderungen eines Signals pro Zeiteinheit).

Übertragungsstörungen

- Übertragungsmedien sind nicht perfekt und daher sind Störungen möglich
 - **Dämpfung** ist der Energieverlust, der bei der Verbreitung eines Signals entsteht;
 - **Laufzeitverzerrung**: Überholen und damit Mischen von Bits
 - **Rauschen**: Beeinträchtigung durch unerwünschte Energie aus anderen Quellen (Kopplung durch benachbarte Drähte)

Man unterscheidet die digitalen Übertragungsverfahren in

- **Analoge Übertragung des Bitstroms**
 - Menge der kodierten Werte ist kontinuierlich, unendlicher Zeichenvorrat, Übertragung als elektrische Schwingung
- **Digitale Übertragung des Bitstroms**
 - Endlicher Zeichenvorrat mit sprungartigem Übergang zwischen den digitalen Zeichen

Sicherungsschicht

Aufgaben

Die Sicherungsschicht (auch Data Link Layer) stellt an einem DL-SAP einen Dienst zur Verfügung, der je nach Schicht-2-Implementierung etwas anders aussieht. Grundsätzlich wird eine zuverlässige Bitübertragung über die Datenverbindung zwischen zwei Rechnersystemen bereitgestellt. Mit „gesichert“ meint man hier, dass überprüft wird, ob die Daten auch beim Empfänger angekommen sind und Übertragungsfehler erkannt und evtl. behoben werden (Fehlersicherung). Man spricht auch von einer gesicherten Ende-zu-Ende-Verbindung zwischen zwei Rechnersystemen.

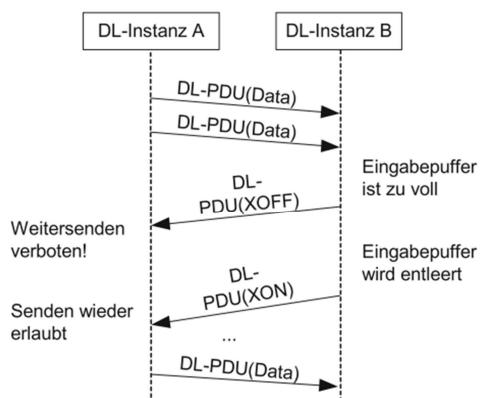
- Gruppierung des übertragenen Bitstroms in logische Einheiten
- Fehlererkennung (Prüfsummen) und ggf. Fehlerkorrektur
- Ende-zu-Ende-Verbindung zwischen Rechnern/Knoten
- Schicht-2-Instanzen verwalten üblicherweise Empfangspuffer und auch Sendepuffer
- Flusskontrolle
- Kanalsteuerung

Für die Aufgabe der Fehlererkennung und evtl. Behebung werden sog. ErrorDetection-Codes, also Prüfsummen, wie etwa Cyclic-Redundancy-Codes (CRC), mitgesendet. Hier handelt es sich im Allgemeinen um redundante Information, die es ermöglicht, gewisse Fehler zu erkennen und bei ausreichender Redundanz sogar automatisch zu beheben. In der Praxis wird allerdings aus Aufwandsgründen selten eine Korrektur vorgenommen, ein erneutes Senden ist hier effizienter. Daten, die ein Sender abschickt, dürfen nicht verloren gehen. Hier kann man in der Sicherungsschicht ebenfalls Maßnahmen ergreifen, wobei folgende Protokollmechanismen in Kombination angewendet werden:

- Bestätigungen senden
- Zeitüberwachungen (Timerüberwachung) für Bestätigungen durchführen
- Sendungswiederholung durchführen
- Erkennen von Duplikaten über Sequenzzähler und Verwerfen dieser

Es gibt unterschiedliche Sicherungsprotokolle. Man spricht hier von verbindungsorientierten und verbindungslosen Protokollen und meint damit, dass entweder vor dem Nutzdatenaustausch eine Verbindung aufgebaut werden muss oder nicht.

Eine weitere Aufgabe der Sicherungsschicht ist die Flusskontrolle zur Vermeidung von Rückstaus und daraus resultierenden Datenverlusten. Unter Flusskontrolle versteht man also die Sicherstellung, dass ein überlasteter Empfänger nicht von einem Sender mit Nachrichten überschwemmt wird, die er nicht verarbeiten kann. Durch geeignete Flusskontrollmechanismen kann man den Sender ausbremsen, wofür es mehrere Möglichkeiten gibt.

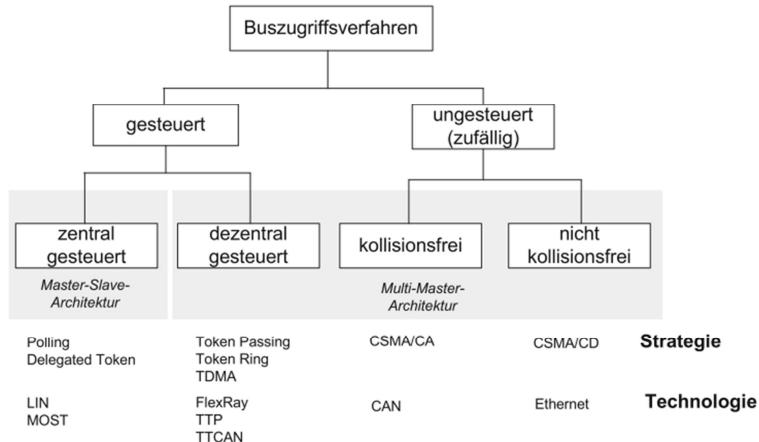


Ein klassisches Protokoll zur Flusssteuerung ist das in Abbildung dargestellte XON/XOFF-Protokoll (auch als Stop-and-Wait-Protokoll bezeichnet). Wird der Puffer zu voll, ist also z. B. eine achzigprozentige Befüllung erreicht, so sendet die DL-Instanz auf Empfängerseite das Zeichen XOFF (ASCII-Zeichen). Die sendende DL-Instanz beendet das Senden, bis es ein XON empfängt. XON signalisiert dem Sender, dass der Empfänger wieder Platz in seinem Puffer hat.

Eine andere Möglichkeit zur Flusskontrolle ist der Sliding-Window-Mechanismus, in dem Sender und Empfänger darüber Buch führen, wie viele Nachrichten gesendet werden dürfen, bevor der Sender ausgebremst wird. Der Empfänger bestätigt ankommende Nachrichten oder Byte über eine ACK-PDU und sendet zusätzlich die Information, wie groß das „Sendefenster“ für eine bestimmte Verbindung noch ist. Einige der Mechanismen aus der Schicht 2, wie das Sliding-Window-Verfahren, finden auch in der Transportschicht Anwendung, gelten dort aber für eine Endezu-Ende-Kommunikation, die evtl. über mehrere Rechner geht und nicht wie in Schicht 2 „nur“ zwischen zwei Rechnersystemen abgewickelt wird.

Buszugriffsverfahren

In Bus-basierten Netzen sind sog. Buszugriffsverfahren für den Medienzugang erforderlich. Dies sind Mechanismen für den zeitgleichen Zugriff auf das Übertragungsmedium durch mehrere Knoten (Stationen). Diese Bus- oder allgemein Kanalzugriffsverfahren sind in der Schicht 2 und zwar in der MAC-Teilschicht angeordnet. Betrachtet werden im Folgenden vorwiegend Buszugriffsverfahren im LAN-Bereich. Die Zuteilung des Busses muss eindeutig geregelt werden. Man kann die Buszugriffsverfahren in gesteuerte und ungesteuerte Verfahren einteilen. Gesteueter Zugriff bedeutet, dass es einen klar definierten Steuerungsmechanismus für die Zuteilung des Busses gibt, während ein ungesteuertes Zugriffsverfahren zufällig ist. Ungesteuerter Zugriff erfordert eine Carrier-Sense-MultipleAccess-Strategie (kurz: CSMA-Strategie)



Man unterscheidet hier wiederum in kollisionsfreien und nicht kollisionsfreien Buszugriff. Bei letzterem kann jeder Knoten zu jeder Zeit auf den Bus zugreifen, was zu Kollisionen führen kann. Daher ist eine Kollisionsstrategie erforderlich.

Strategien, die Kollisionen zwar erkennen und behandeln, aber nicht vermeiden, werden als CSMA/Collision-Detecting-Strategien (kurz: CSMA/CD) bezeichnet. Der bekannteste Technologievertreter dieser Strategie ist Ethernet. Eine Strategie, die Kollisionen möglichst vermeiden kann, wird als CSMA/Collision-Avoidance-Strategie (kurz: CSMA/CA) bezeichnet.

CSMA Protokolle

Non-persistent CSMA: Bei non-persistent CSMA wird vor dem Senden geprüft, ob der Kanal frei ist. Falls dies der Fall ist, wird gesendet. Ist dies nicht der Fall, wird eine zufällig verteilte Zeit gewartet, und dann wieder von vorne begonnen.

p-persistent CSMA: Beim p-persistent CSMA prüft eine Station vor dem Senden auch, ob der Kanal frei ist. Falls dies der Fall ist, wird das Paket nur mit der Wahrscheinlichkeit p sofort gesendet. Mit der Wahrscheinlichkeit $1 - p$ wird zunächst eine bestimmte Zeit gewartet und danach der Sendevorschuss wiederholt.

1-persistent CSMA: Eine Spezialvariante ist das 1-persistent CSMA. In dieser Variante hört eine Station den Kanal ab, wenn sie senden möchte. Ist der Kanal besetzt, wird gewartet, bis er frei ist, ansonsten wird ein Rahmen übertragen. Der Name kommt daher, weil eine Station mit einer Wahrscheinlichkeit von 1 sendet, wenn der Kanal frei ist.

Andere Klassifizierung von Buszugriffsverfahren:

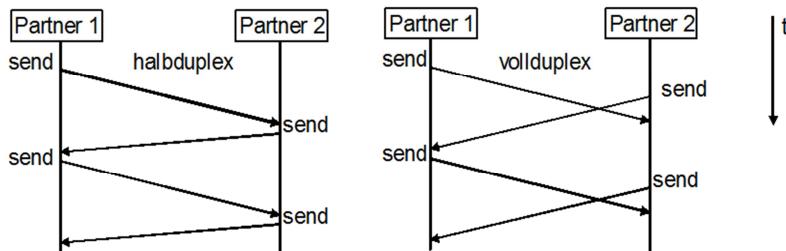
- Wettkampfverfahren
- Token-Passing-Verfahren
- Distributed-Queue-Dual-Bus-Verfahren (DQDB)

Transportzugriff

Konzepte:

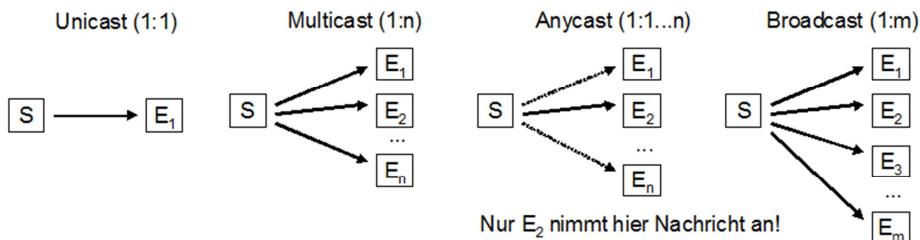
Halb- und vollduplex

- **Halbduplex:** Nur einer der Partner sendet zu einer Zeit
- **Vollduplex:** beide Partner können unabhängig voneinander senden



Empfängeradressierung

- Unicast: Nur ein Empfänger wird adressiert
- Alle anderen Varianten adressieren mehrere Empfänger

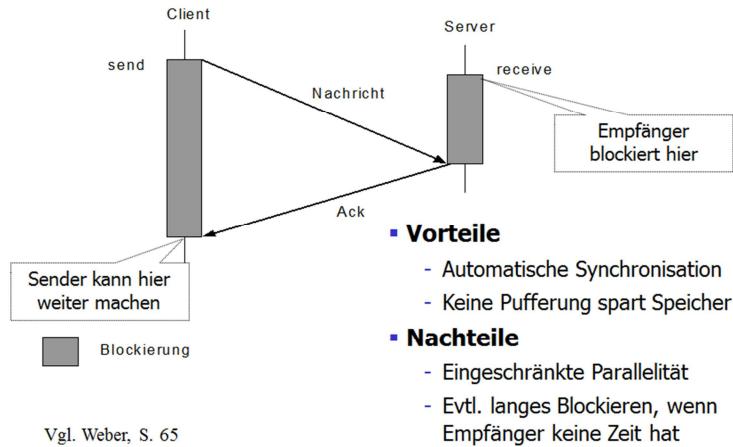


Blockierung

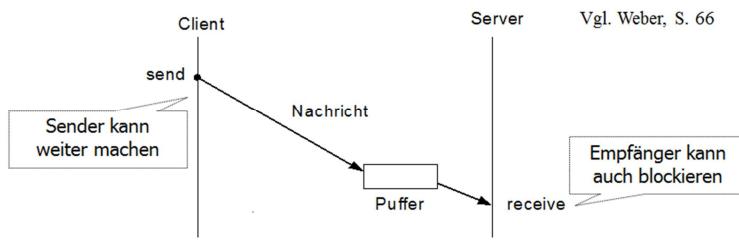
- Beim Nachrichtenaustausch unterscheidet man:
 - Synchronie Kommunikation
 - Asynchrone Kommunikation
- Synchron bedeutet:
 - **Blockierend**
 - Der Sender wartet, bis eine Methode send **mit einem Ergebnis** zurückkehrt
- Asynchron bedeutet:
 - **Nicht blockierend**
 - Der Sender kann weiter machen, wenn die Nachricht mit einer Methode send **in einen Transportpuffer** gelegt wurde

Pufferung von Nachrichten

- Puffer für ankommende Nachrichten werden **in den Protokollinstanzen** (meist im Betriebssystemkern) verwaltet
- Die Instanzen **kopieren** die Nachrichten in den Adressraum der empfangenden Anwendungsprozesse
- Pufferspeicher müssen **verwaltet** werden (→ Overhead)
- Pufferspeicher **benötigen Adressraum** (Speicher)
- Pufferspeicher sind **begrenzt** (→ evtl. Verwerfen von Nachrichten, wenn sie voll sind)
Blockierung Synchronie Kommunikation



Blockierung Asynchrone Kommunikation



■ Vorteile

- Zeitliche Entkopplung
- Bessere Parallelarbeit möglich
- Ereignisgesteuerte Kommunikation möglich

■ Nachteile

- Zwischenpufferung notwendig
- Puffer voll führt trotzdem zum Blockieren wegen gesicherter Übertragung

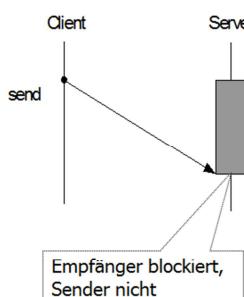
Kommunikationsformen

- Man unterscheidet weiterhin
 - **Meldungsorientierte Kommunikation**
 - Einwegnachrichten ohne Antwort
 - **Auftragsorientierte Kommunikation**
 - Request und Response (**mit Ergebnis**)
 - Entfernter Dienstauftruf

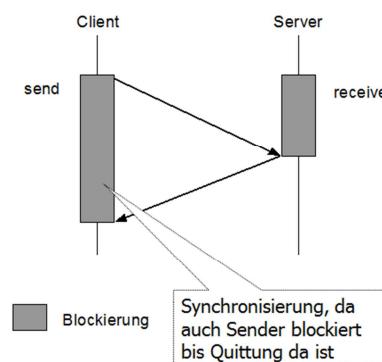
	asynchron	synchron
meldungsorientiert	Datagramm	Rendezvous
auftragsorientiert	asynchroner entfernter Dienstauftruf	synchroner entfernter Dienstauftruf

Meldungsorientierte Kommunikation

Datagramm



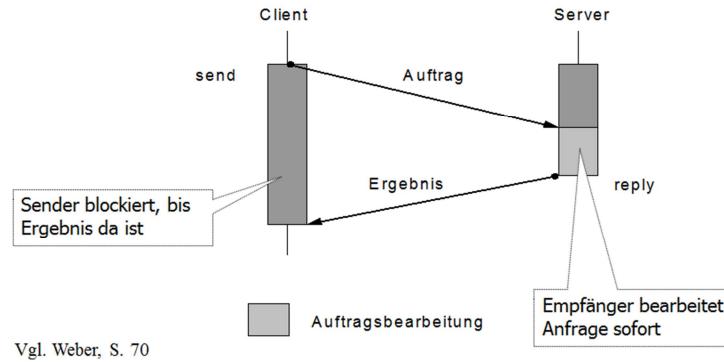
Rendezvous



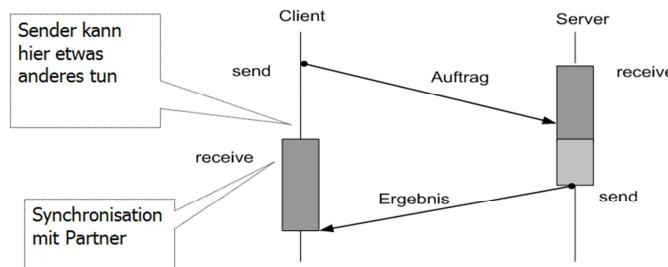
Vgl. Weber, S. 69

Auftragsorientierte Kommunikation

- Synchroner entfernter Dienstaufruf
 - Mit Rendezvous verwandt



- Asynchroner entfernter Dienstaufruf



Fehlersemantiken

- Es gibt viele Fehlerursachen
 - Netzwerkfehler
 - Sender fällt vor Empfang des Ergebnisses aus (\rightarrow verwaiste Aufträge, Orphans)
 - Empfänger fällt während der Bearbeitung eines Requests aus
 - ...
 - Ein Ausfall ist zu jeder Zeit möglich
 - Das Kommunikationssystem kann sich hier je nach Realisierung unterschiedlich verhalten
- Verschiedene Fehlersemantiken möglich

- Lokal: Alles fällt komplett aus oder es läuft
- Verteilt: Verschiedene Ausfallsituationen zu betrachten

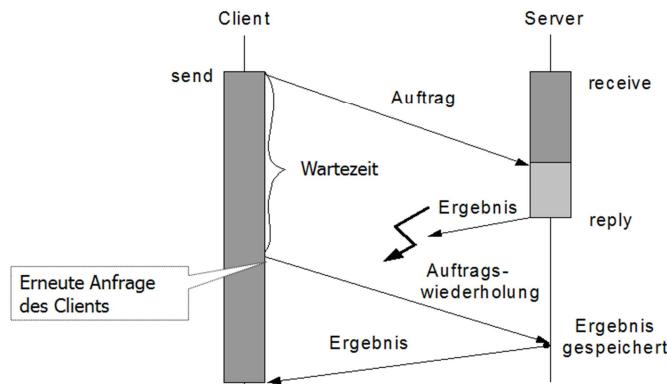
Fehlerklasse	Fehlerarten			
	Fehlerfreier Ablauf	Nachrichtenverlust	Ausfall des Servers	Ausfall des Clients
Maybe	Ausführung: 1 Ergebnis: 1	Ausführung: 0/1 Ergebnis: 0	Ausführung: 0/1 Ergebnis: 0	Ausführung: 0/1 Ergebnis: 0/1
At-Least-Once	Ausführung: 1 Ergebnis: 1	Ausführung: ≥ 1 Ergebnis: ≥ 1	Ausführung: ≥ 0 Ergebnis: ≥ 0	Ausführung: ≥ 0 Ergebnis: 0
At-Most-Once	Ausführung: 1 Ergebnis: 1	Ausführung: 1 Ergebnis: 1	Ausführung: 0/1 Ergebnis: 0/1	Ausführung: 0/1 Ergebnis: 0
Exactly-Once	Ausführung: 1 Ergebnis: 1	Ausführung: 1 Ergebnis: 1	Ausführung: 1 Ergebnis: 1	Ausführung: 1 Ergebnis: 1

Nach Schill (2007), Verteilte Systeme

Wiederanlauf und Rücksetzmechanismen vorhanden → Transaktionen

- Beispiel für eine Fehlersituation mit entsprechender Reaktion
 - Der Client erhält Antwort vom Server nicht und reagiert mit einem Timeout
 - Die Antwort des Servers ging verloren
 - Der Server speichert die Antwort
 - Der Client wiederholt den Auftrag nach dem Timeout
 - Der Server kann nun das gespeicherte Ergebnis senden
- Implementierungsaufwand!

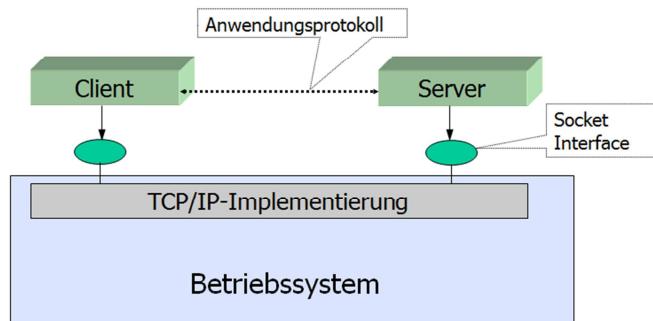
▪ Möglicher Ablauf im Beispiel



Überblick über Sockets

- Die Socket-Schnittstelle ist eine API, mit der man Kommunikationsanwendungen entwickeln kann
- Die Socket-Schnittstelle ist eine **Transportzugriffs-schnittstelle**
- Sockets wurden in der Universität von Berkeley entwickelt (BSD-Version von UNIX), **erste Version 4.1cBSD-System** für die VAX aus dem Jahre 1982
- Die **Originalversion** der Socket-Schnittstelle stammt von Mitarbeitern der **Firma BBN** (ARPA-Projekt, 1981)
- Sockets sind heute der **De-facto-Standard**
- Die Socket API **unterstützt** vor allem **Client-Server-Anwendungen**, was aus dem Programmiermodell hervorgeht:
 - Aktiver Partner = Client
 - Passiver Partner = Server
- Sockets sind **Kommunikationsendpunkte** innerhalb der Applikationen, die in der Initialisierungsphase miteinander verbunden werden
- Dabei spielt es keine Rolle, auf welchen Rechnern die miteinander kommunizierenden Prozesse laufen
- Die TCP-Socket-Schnittstelle ist **stream-orientiert**: Es wird ein Datenstrom eingerichtet
- Die UDP-Socket-Schnittstelle ist nachrichten-orientiert
- Es ist **verbindungsorientierte** (TCP-basierte) und **verbindungslose** (UDP-basierte) Kommunikation möglich
- Die Adressierung der Kommunikationspartner erfolgt über die Kommunikationsendpunkte über das Tupel (IP-Adresse, Portnummer)

- Systemeinbettung von Sockets



Phasen der Kommunikation über Sockets

- Die Verwendung von Sockets in einer Applikation erfolgt programmiertechnisch in **3 Phasen**:
 1. Initialisierungsphase
 - Diese Phase ist für UDP symmetrisch, d.h. es gibt keine Unterscheidung zwischen Sender- und Empfänger-Sockets
 - Für TCP ist diese Phase asymmetrisch zwischen Client- und Server-Socket, d.h. ein Server-Socket wird anders initialisiert als ein Client-Socket
 2. Lese- und Schreibphase
 - Sowohl für UDP als auch für TCP ist diese Phase symmetrisch, d.h. jeder Socket kann gleichermaßen senden und empfangen
 3. Aufräumphase
 - Die benötigten Ressourcen werden freigegeben
 - Diese Phase ist sowohl für TCP als auch für UDP symmetrisch

UDP-Sockets allgemein

- Paket-basierte Kommunikation im Gegensatz zu Streams-basiert
- Keine Empfangsgarantie
- Keine Ordnung der Pakete
- Duplikate sind möglich
- Vorteile:
 - Separate Nachrichten können gesendet werden
 - Ein Fehler in einer Nachricht wirkt sich nicht auf die folgenden Nachrichten aus
 - Es werden immer nur ganze Nachrichten empfangen

Zusammenfassung Sockets:

- Wie andere Programmiersprachen bietet Java Sockets für die TCP/IP- bzw. UDP/IP-Kommunikation an
- Socketadressen bestehen aus zwei Komponenten, der IP-Adresse bzw. dem logischen DNS-Namen des betreffenden Rechners und einer Portnummer
- Für TCP-Verbindungen gibt es Server-Sockets und Client-Sockets:
 - Wenn sich ein Client-Socket an einen Server-Socket anbindet, wird auf Server-Seite ein neuer Socket erzeugt, der den Endpunkt dieser Verbindung darstellt
 - Das eigentliche Senden und Empfangen von Daten geschieht über Schreib- und Leseströme, die man mit einem Socket assoziieren kann

- UDP-Sockets = Endpunkte zum Senden und Empfangen von Datagrammen
 - UDP ist verbindungslos
 - UDP ist nicht zuverlässig
 - UDP garantiert keine Reihenfolgetreue
- Das Anwendungsprogramm muss diese Dinge regeln
- Bei Nutzung von UDP-Sockets sollte man berücksichtigen:
 - 16 Bits für Paketlänge, also Datenlänge begrenzen, kein Stream
 - Vermeidung von Fragmentierung wichtig, wenn ein Fragment (IP) verloren geht, wird die ganze Nachricht verworfen

Transportschicht Grundlagen

Verbindungen

- Man unterscheidet
 - **verbindungsorientierte** Transportdienste
 - **verbindungslose** Transportdienste

Verbindungsorientierte Transportdienste sind durch drei Phasen gekennzeichnet:

- Verbindungsaufbau
- Datenübertragung
- Verbindungsabbau

Der Sender adressiert den Empfänger beim Verbindungsaufbau und sendet beim Datenaustausch in den T-PDUs (auch als Segmente bezeichnet) nur noch seine Verbindungs-Identifikation mit. Für eine Verbindung muss bei beiden Kommunikationspartnern in den Transportinstanzen ein gemeinsamer Verbindungskontext verwaltet werden, der sich für jedes Protokoll in einem Zustandsautomaten beschreiben lässt.

Verbindungslose Dienste sind meist durch folgende Eigenschaften charakterisiert:

- Der Verlust von Datenpaketen ist möglich
- Die Daten können ggf. verfälscht werden
- Die Reihenfolge ist nicht garantiert
- Die Adressierungsinformation muss in allen T-PDUs enthalten sein

Verbindungsorientierte Protokolle sind komplexer und daher aufwändiger zu implementieren, bieten aber in der Regel eine höhere Zuverlässigkeit und garantieren meist eine fehlerfreie und reihenfolgerichtige Auslieferung der Daten beim Empfänger

Protokollfunktionen in Transportprotokollen

- Verbindungsmanagement und Adressierung
- Zuverlässiger Datentransfer
- Flusskontrolle
- Staukontrolle
- Multiplexierung (Multiplexing) und Demultiplexing
- Fragmentierung und Defragmentierung

Verbindungsmanagement und Adressierung

Die Anwendungsprozesse sind über die zugehörigen T-SAPs adressierbar, während die Transportschicht N-SAPs zur Adressierung der Rechnersysteme nutzt. Eine T-Instanz unterstützt in der Regel mehrere T-SAPs. Transportadressen sind meist sehr kryptisch, weshalb oft symbolische Adressen benutzt werden, die über sog. Naming-Services (Directory Services) auf Transportadressen abgebildet werden, ohne dass der Anwendungsprogrammierer diese kennen muss. Ein Beispiel für einen klassischen Naming-Service ist das im Internet unbedingt erforderliche DNS (Domain Name System).

Verbindungsorientierte Dienste erfordern einen Verbindungsaufbau, in dem zunächst auf beiden Seiten ein Verbindungskontext aufgebaut wird. Die Endpunkte der Verbindung werden im ISO/OSI-Jargon auch als Connection End Points (CEP) bezeichnet.

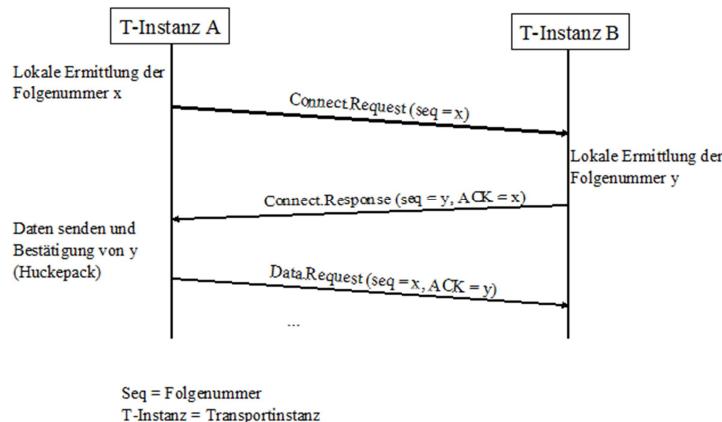
Verbindungsauflauf

- **Zwei-Wege-Handshake-Protokolle**
- **Drei-Wege-Handshake-Protokolle**
- **Vorsicht Duplikate!**
 - Diverse Fehlerszenarien möglich
 - Mechanismus der **Folgenummern** kombiniert mit einer maximalen Paketlebensdauer
 - Folgenummern sind einfache Zähler

Verbindungsauflauf, Drei-Wege-Handshake

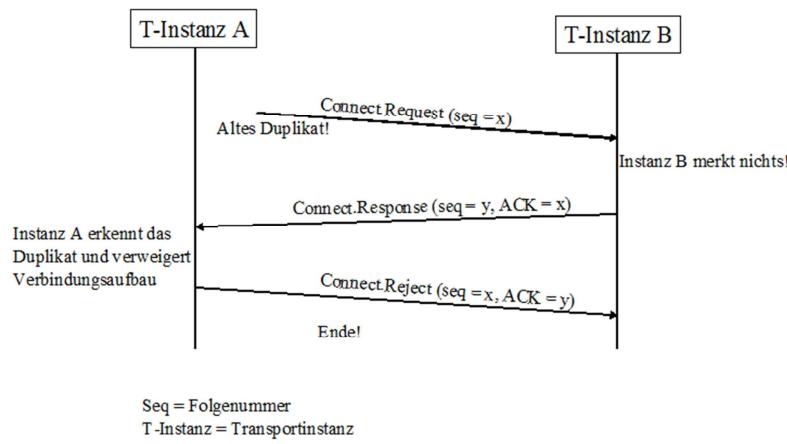
**Normaler
Protokollverlauf**

**Instanz A und B suchen eigene
Folgenummern x und y (seq) aus**

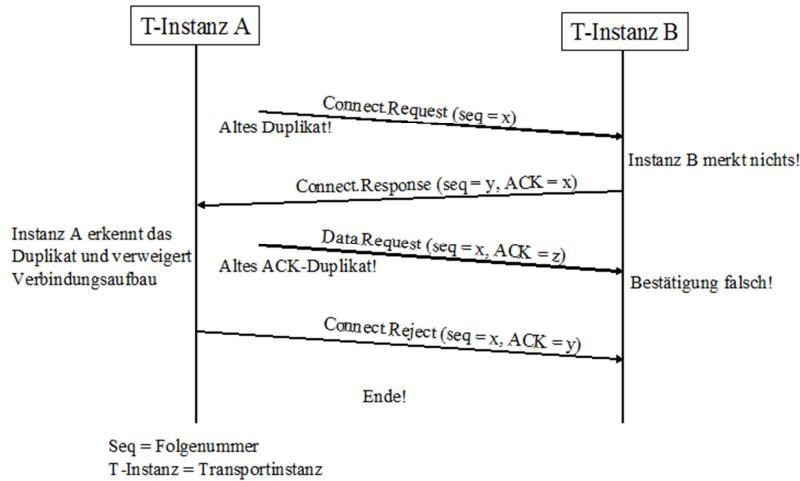


Altes CR-Duplikat

taucht auf



Duplikat von Connect-Request und Duplikat von ACK tauchen plötzlich auf



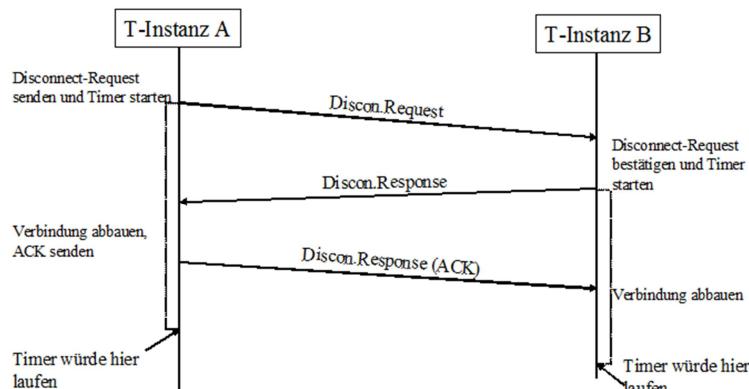
Verbindungsabbau

- Anforderung:
 - Beim Verbindungsabbau dürfen keine Nachrichten verloren gehen
- Datenverlust kann vorkommen, wenn
 - eine Seite einen Verbindungsabbau initiiert,
 - die andere aber vor Erhalt der Disconnect-Request-PDU noch eine Nachricht sendet
 - Diese **Nachricht ist verloren (Datenverlust)**
- Anspruchsvolles Verbindungsabbau-Protokoll notwendig:
 - Dreiwege-Handshake-Mechanismus wird auch hier genutzt
 - Beide Seiten bauen ihre „Senderichtung“ ab

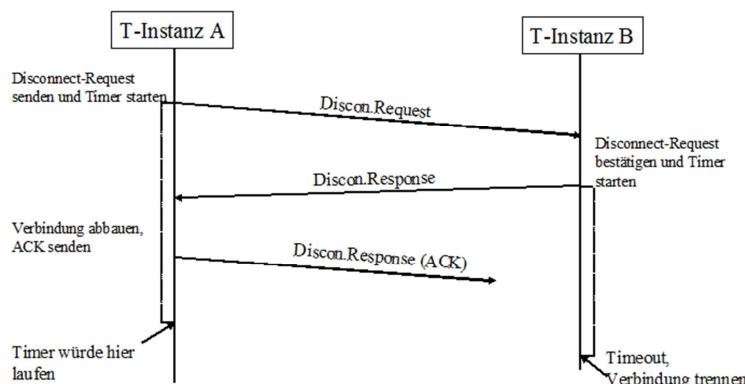
Verbindungsabbau, Timerüberwachung

- Kein Protokoll ist absolut zuverlässig
- Es wird immer eine Seite geben, die **unsicher** ist, ob die letzte Nachricht angekommen ist
- Übertragen auf den Verbindungsabbau:
 - Beim Dreiwege-Handshake kann **immer** ein Disconnect-Request oder **eine Bestätigung verloren gehen**
- Praktische Lösung: **Timerüberwachung** mit begrenzter Anzahl an Nachrichtenwiederholungen
- Nicht unfehlbar, aber doch ganz zufriedenstellend

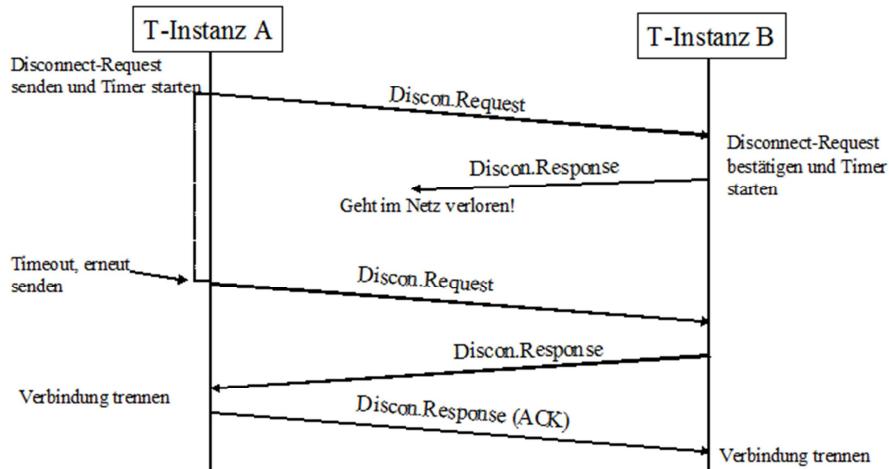
Timerüberwachung beim Verbindungsabbau Szenario „Normaler Ablauf“



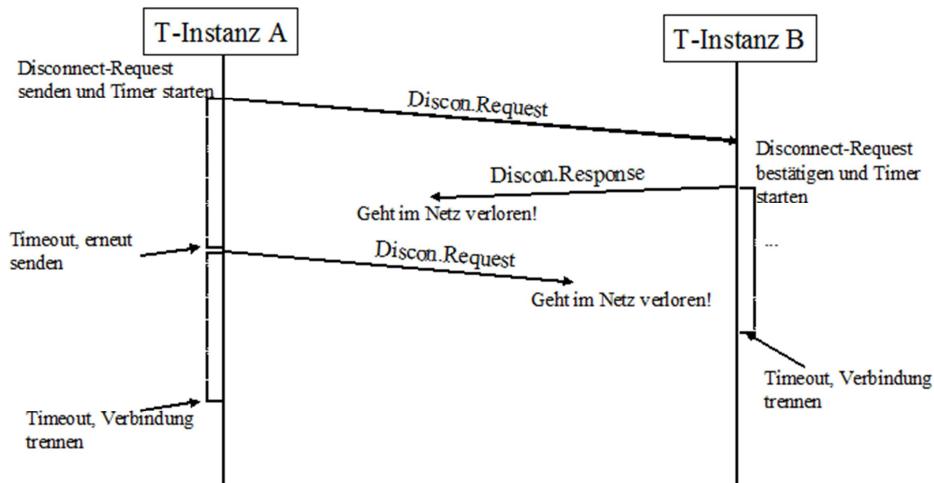
Szenario „Timer läuft ab“



Szenario „Disconnect-Response geht verloren“



Szenario „Zwei Disconnect-PDUs gehen verloren“



Zuverlässiger Datentransfer

- Was heißt hier zuverlässige Datenübertragung?
 - Garantierte Ausführung ? → Nein!
 - Transaktionssicherheit → Nein!
 - Sicherstellen der Übertragung durch Quittierung und Übertragungswiederholung → Ja!
- Quittierungsvarianten
 - Positiv selektiv
 - Positiv kumulativ
 - Negativ selektiv
 - Kombination der Verfahren möglich
- Varianten der Übertragungswiederholung
 - Selektiv
 - Go-back-N

Quittierungsverfahren

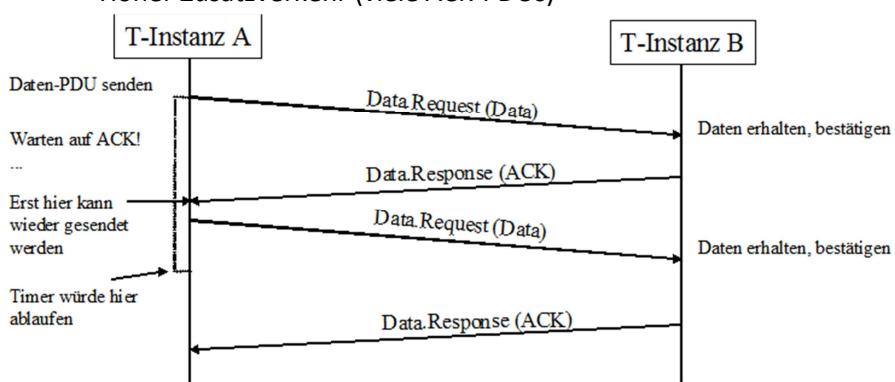
Um sicher zu sein, dass Nachrichten in einem verlustbehafteten Kanal richtig ankommen, sind Quittierungsverfahren erforderlich. Hier gibt es gravierende Unterschiede hinsichtlich der Leistung. Einfache Verfahren quittieren jede Data-PDU mit einer ACK-PDU (Acknowledge), was natürlich nicht sehr leistungsfähig ist. Der Sender muss bei diesem Stop-and-Wait-Protokoll immer warten, bis eine Bestätigung eintrifft, bevor er weitere Data-PDUs senden kann. Aufwändiger Verfahren ermöglichen das Senden mehrerer Daten und sammeln die Bestätigungen ggf. kumuliert ein. Hier spricht man auch von Pipelining.

Man unterscheidet bei Quittungsverfahren:

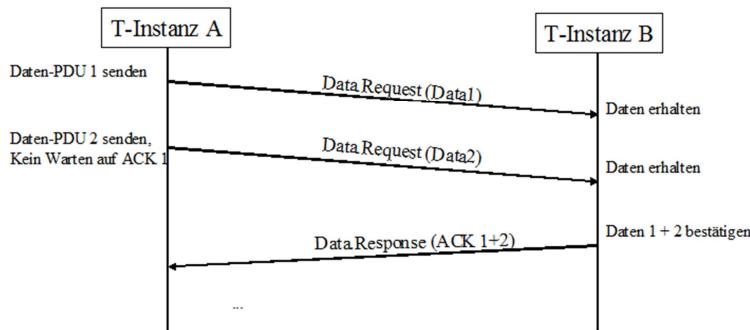
- Positiv-selektives Quittierungsverfahren wie das Stop-and-Wait-Protokoll:
Hier wird vom Empfänger eine Quittung (ACK) pro empfangener Nachricht gesendet. Dies hat einen hohen zusätzlichen Nachrichtenverkehr zur Folge.
- Positiv-kumulatives Quittierungsverfahren: Eine Quittung wird für mehrere Nachrichten verwendet. Diese Reduzierung der Netzlast hat aber den Nachteil, dass Information über einen Datenverlust verspätet an den Sender übertragen wird.
- Negativ-selektives Quittierungsverfahren: Es werden vom Empfänger nur selektiv nicht empfangene, also verlorengegangene, Nachrichten erneut vom Sender angefordert. Alle Nachrichten, bei denen keine Nachfrage kommt, gelten beim Sender als angekommen. Dieses Verfahren reduziert die Netzlast weiter. Ein Verlust von negativen Quittungen (Nachfragen des Empängers) ist jedoch möglich und muss behandelt werden.
- Eine Kombination der Verfahren: In Hochleistungsnetzen verwendet man z. B. das positiv kumulative kombiniert mit dem negativ selektiven Verfahren.

▪ **Positiv selektives** Quittierungsverfahren

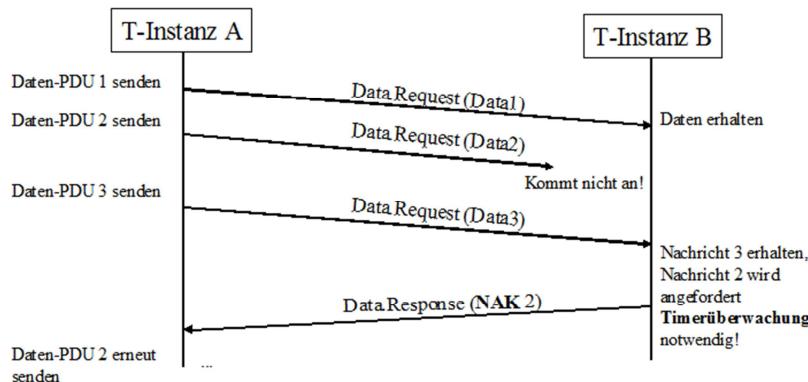
- Stop-and-Go (Stop-and-Wait): Eine Quittung pro gesendeter Nachricht
- Hoher Zusatzverkehr (viele ACK-PDUs)



- **Positiv kumulatives** Quittierungsverfahren
 - Eine Quittung für mehrere Nachrichten
 - Reduzierung der Netzlast
 - Nachteil: Verspätete Information an den Sender über Datenverlust



- **Negativ selektives** Quittierungsverfahren
 - Weitere Reduzierung der Netzlast
 - Problem: Verlust von Quittungen und dessen Behandlung



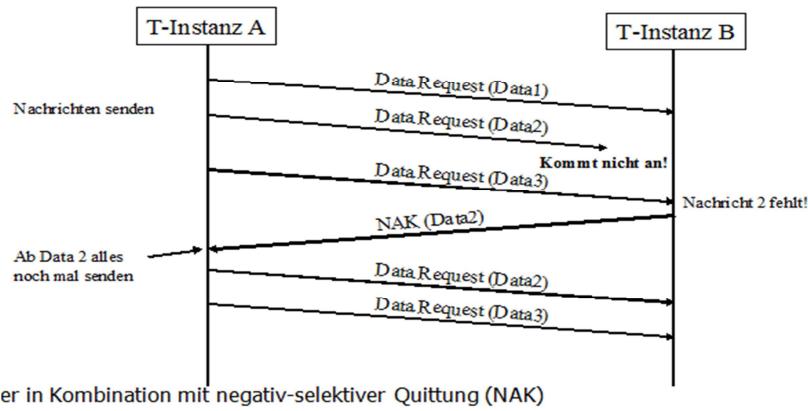
Übertragungswiederholung

Die Übertragungswiederholung ist dabei auf zwei Arten üblich. Entweder man verwendet eine selektive Wiederholung oder ein Go-Back-N-Verfahren:

- Beim selektiven Verfahren werden nur die negativ quittierten Nachrichten wiederholt. Der Empfänger puffert die nachfolgenden Nachrichten, bis die fehlende da ist. Erst wenn dies der Fall ist, werden die Daten am T-SAP nach oben zur nächsten Schicht weitergereicht. Nachteilig ist dabei, dass eine hohe Pufferkapazität beim Empfänger erforderlich ist. Von Vorteil ist, dass die reguläre Übertragung während der Wiederholung fortgesetzt werden kann.
- Beim Go-Back-N-Verfahren werden die fehlerhafte Nachricht sowie alle nachfolgenden Nachrichten erneut übertragen. Nachteilig ist hier, dass die reguläre Übertragung unterbrochen wird. Von Vorteil ist, dass beim Empfänger nur geringe Speicherkapazität erforderlich ist.

Wiederholungsvarianten

- **Go-Back-N**
 - Übertragungswiederholung der fehlerhaften Nachricht sowie aller nachfolgenden
 - Die reguläre Übertragung wird unterbrochen
 - Vorteil: Geringe Speicherkapazität beim Empfänger



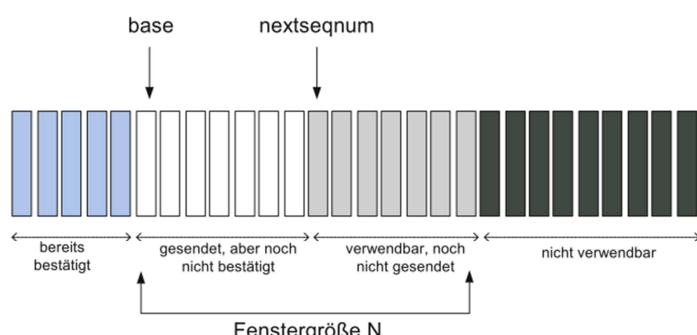
Flusskontrolle

- Steuerung des Datenflusses
- Vermeidet **Überlastung des Empfängers**
- Traditionelle Verfahren sind:
 - **Stop-and-Go (Stop-and-Wait)**
 - Einfachstes Verfahren
 - Kopplung von Fluss- und Fehlerkontrolle
 - Nächste Nachricht wird erst nach der Quittierung gesendet
 - **Fensterbasierte Flusskontrolle**
 - Empfänger vergibt sog. **Sendekredit**, also eine max. Menge an Nachrichten oder Bytes, die unquittiert an ihn gesendet werden dürfen
 - Empfänger kann den Sendekredit durch **positive Quittungen** erhöhen
 - Vorteil: Kontinuierlicher Datenfluss und höherer Durchsatz als bei Stop-and-Go möglich

Für die fensterbasierte Flusskontrolle werden in den Transportinstanzen für jeden

Kommunikationspartner vier Folgenummern-Intervalle verwaltet sind:

- Das linke Intervall sind Folgenummern, die gesendet und vom Empfänger bereits bestätigt wurden.
- Das zweite Intervall von links stellt alle Folgenummern dar, die gesendet, aber vom Empfänger noch nicht bestätigt wurden. Der Zeiger base verweist auf den Anfang dieses Intervalls.
- Das nächste Intervall gibt alle Folgenummern an, die noch verwendet werden dürfen, ohne dass eine weitere Bestätigung vom Empfänger eintrifft. Der Zeiger nextseqnum zeigt auf den Anfang des Intervalls.
- Das vierte Intervall zeigt die Folgenummern, die noch nicht verwendet werden dürfen.

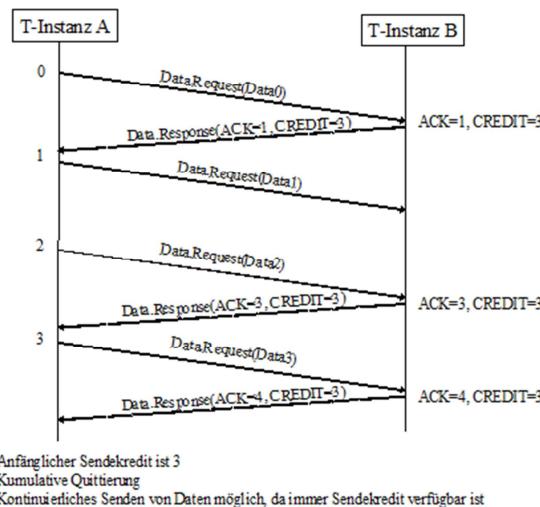


Die beiden inneren Intervalle bilden zusammen das aktuelle Fenster, geben also den verfügbaren Sendekredit an. Trifft nun eine Bestätigung des Empfängers ein, wandert das Fenster um die Anzahl der bestätigten Folgenummern nach rechts, d.h. der Zeiger base wird nach rechts verschoben. Gleichzeitig wird eine entsprechende Anzahl an Folgenummern aus den bisher nicht verwendbaren Folgenummern dem dritten Intervall zugeordnet. Der Zeiger nextseqnum wird nicht verändert. Erst wenn wieder Nachrichten gesendet werden, wird dieser entsprechend der Anzahl an benutzten Folgenummern nach rechts verschoben.

- Fensterbasierte Flusskontrolle in langsamem Netzen

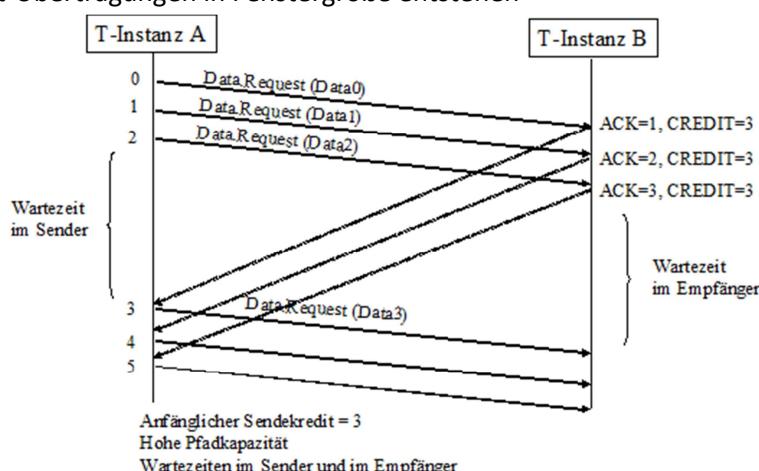
- Geringe Pfadkapazität in langsamem Netzen
- Instanz A kann kontinuierlich Daten senden
- Sendekredit immer verfügbar

Pfadkapazität =
Bandbreiten-Verzögerungs-Produkt



- Fensterbasierte Flusskontrolle bei hoher Pfadkapazität

- Instanz A hat bei kleinen Krediten häufig keinen Sendekredit → Wartezeiten beim Empfänger
- Burst-Übertragungen in Fenstergröße entstehen



Staukontrolle (Überlastkontrolle, Congestion Control)

- Nicht mit Flusskontrolle verwechseln
- Durch Staukontrolle sollen **Verstopfungen** bzw. Überlastungen im Netz **vermieden** werden
- Staukontrolle in der Transportschicht durch **Ende-zu-Ende-Steuerung** zwischen Endsystemen
- Staukontrolle ist ein Mechanismus mit **netzglobalen Auswirkungen**
- Beispiel später: Slow-Start-Verfahren bei TCP

Zusammenfassung zur Transportschicht

- Logischer Ende-zu-Ende-Transport
- Verbindungslos versus verbindungsorientiert
- Verbindungsmanagement und Adressierung
 - Verbindungsaufbau
 - Verbindungsabbau
- Zuverlässiger Datentransfer
 - Quittierungsverfahren
 - Übertragungswiederholung
- Flusskontrolle
- Staukontrolle

TCP und UDP

TCP ist ein Transportprotokoll und gibt der TCP/IP-Protokollfamilie seinen Namen. Es ermöglicht eine Ende-zu-Ende-Beziehung zwischen kommunizierenden Anwendunginstanzen. TCP ist sehr weit verbreitet und als Industrienorm akzeptiert. Es ist ein offenes, frei verfügbares Protokoll und damit nicht an einen Hersteller gebunden. Neben UDP ist TCP das Transportprotokoll im Internet, auf das die meisten Anwendungen basieren. Als Transportzugriffsschnittstelle dient die Socket-Schnittstelle. Ein T-SAP wird in TCP durch einen Port mit einer Portnummer identifiziert. Ein Anwendungsprozess benötigt also einen lokalen TCP-Port und kommuniziert über diesen mit einem anderen Anwendungsprozess, der ebenfalls über einen TCP-Port adressierbar ist.

TCP ist ein verbindungsorientiertes, zuverlässiges Protokoll. Die Protokollmechanismen von TCP werden in diesem Abschnitt beschrieben. Grob gesagt kümmert sich TCP um die Erzeugung und Erhaltung einer gesicherten Ende-zu-Ende-Verbindung zwischen zwei Anwendungsprozessen auf Basis von IP. TCP garantiert weiterhin die Reihenfolge der Nachrichten und die vollständige Auslieferung, übernimmt die Fluss- und Staukontrolle und kümmert sich um die Multiplexierung sowie die Segmentierung.

TCP stellt also sicher, dass Daten

- nicht verändert werden,
- nicht verloren gehen,
- nicht dupliziert werden und
- in der richtigen Reihenfolge eintreffen.

TCP ermöglicht die Kommunikation über vollduplex-fähige, bidirektionale virtuelle Verbindungen zwischen Anwendungsprozessen. Das bedeutet, beide Kommunikationspartner können zu beliebigen Zeiten Nachrichten senden, auch gleichzeitig.

Maßnahmen zur Sicherung der Übertragung sind die Nutzung von Prüfsummen, Bestätigungen, Zeitüberwachungsmechanismen mit verschiedenen Timern, Nachrichtenwiederholung, Sequenznummern für die Reihenfolgeüberwachung und das Sliding Windows Prinzip zur Flusskontrolle.

Aufgaben im Detail

- Schaffung einer gesicherten Ende-zu-Ende-Verbindung auf Basis von IP
- Reihenfolgegarantie
- Garantierte Auslieferung
- Staukontrolle
- Flusskontrolle (Vermeidung von Überschwemmungen langsamer Empfänger)
- Multiplexing und Demultiplexing der IP-Verbindung
- Fragmentierung und Defragmentierung der Nachrichten (Segmente!)

Einordnung und Aufgaben

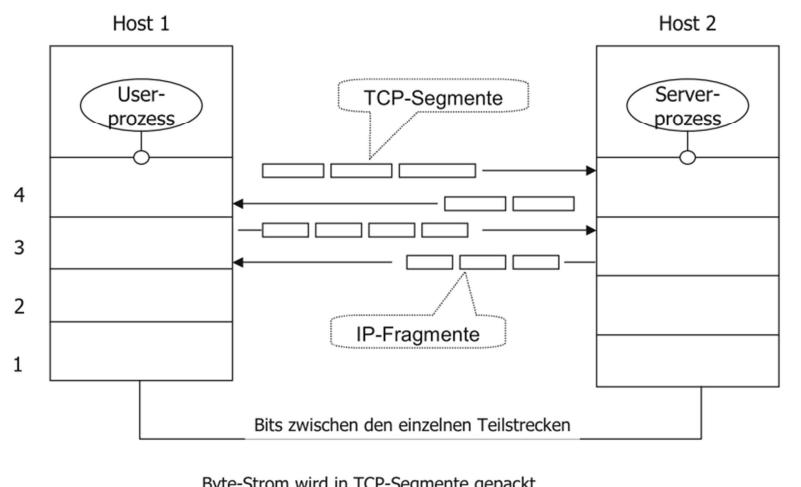
- **Maßnahmen zur Sicherung der Übertragung:**
 - Drei-Wege-Handshake für Verbindungsmanagement
 - Prüfsumme
 - Quittierung (ACK-PDU): Positiv-kumulativ und implizites NAK
 - Zeitüberwachung für jedes Segment (Timer) und Nachrichtenwiederholung
 - Go-Back-N für die Nachrichtenwiederholung
 - Sequenznummern = Folgenummern für die Reihenfolgeüberwachung
 - Sliding Windows Prinzip zur Flusskontrolle
 - Slow-Start-Verfahren

Dienste, Ports und Adressierung

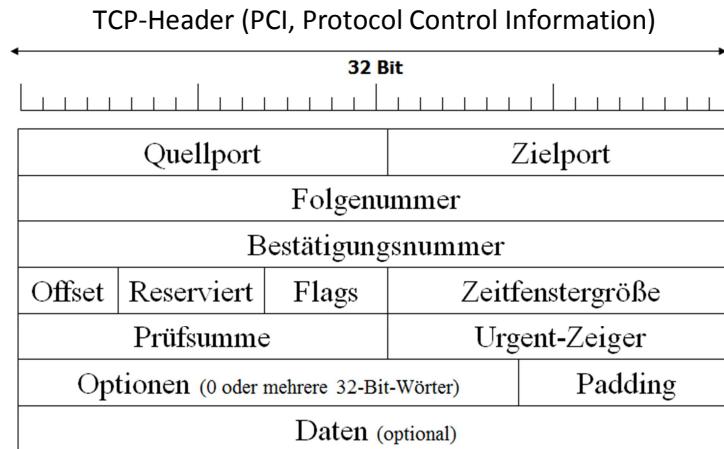
- Anwendungsprozess kommuniziert über eine Adresse, die als **Socket** bezeichnet wird
 - Tupel der Form (IP-Adresse, TCP-Portnummer)
- **Well-known Ports**
 - 16-Bit-Integer
 - Es gibt hier eine Reihe von sog. well-known Ports für reservierte Services (Portnr. <= 1024)
 - Jeder Dienst hat eine eigene Portnummer (siehe Datei /etc/services unter Unix)
- Ein Anwendungsprozess kann auch mehrere Verbindungen unterhalten
- Client-Server-Prinzip leicht realisierbar:
 - Server wartet an einem Port auf Connect-Requests

TCP Header

TCP sieht den Datenstrom (Stream) als eine Sequenz von Octets (Byte) und unterteilt diese zur Übertragung in Segmente. Ein Segment besteht aus einem mind. 20 Byte langen TCP-Header. Die Begrenzung der Segmentlänge für jede Teilstrecke ist aber durch die MTU (Maximum Transfer Unit) für jede Teilstrecke eines Netzes gegeben.



Im Anschluss an den Header werden bei TCP-Data-PDUs die Daten übertragen.



- **Quell- und Zielport**
 - Portnummer des Anwendungsprogramms des Senders und des Empfängers
- **Folgenummer**
 - Nächstes Byte innerhalb des TCP-Streams ($\text{mod } 2^{32}$)
- **Bestätigungsnummer**
 - Gibt das als nächstes erwartete Byte im TCP-Strom an und bestätigt damit den Empfang der vorhergehenden Bytes
- **Offset**
 - Gibt die Länge des TCP-Headers in 32-Bit-Worten an
- **Reserviert**
 - Hat noch keine Verwendung
- **Flags:** Hier handelt es sich um Kontroll-Bits mit unterschiedlicher Bedeutung:

Flag	Bedeutung
URG	Urgent-Zeiger-Feld ist gefüllt
ACK	Bestätigung (z.B. bei Verbindungsaufbau genutzt), d.h. die ACK-Nummer hat einen gültigen Wert
PSH	Zeigt Push-Daten an, Daten dürfen beim Empfänger nicht zwischengespeichert werden, sondern sind sofort an den Empfängerprozess weiter zu leiten
RST	Dient zum <ul style="list-style-type: none"> - Rücksetzen der Verbindung (sinnvoll z.B. bei Absturz eines Hosts) - Abweisen eines Verbindungsaufbauwunsches - Abweisen eines ungültigen Segments
SYN	Wird genutzt beim Verbindungsaufbau
FIN	Wird genutzt beim Verbindungsabbau

- **Zeitfenstergröße**
 - Erlaubt dem Empfänger, mit ACK dem Sender den vorhandenen Pufferplatz in Byte zum Empfang der Daten mitzuteilen
- **Urgent-Zeiger**
 - beschreibt die Position (Byteversatz von der aktuellen Folgenummer ab) an der dringliche Daten vorgefunden
 - Diese Daten werden vorrangig behandelt
 - Selten genutzt!
- **Prüfsumme**
 - Verifiziert das Gesamtpaket (Header+Daten) auf Basis eines einfachen Prüfsummenalgorithmus
 - Einer-Kompliment der ganzen Nachricht inkl. Header und Pseudo-Header (siehe UDP)

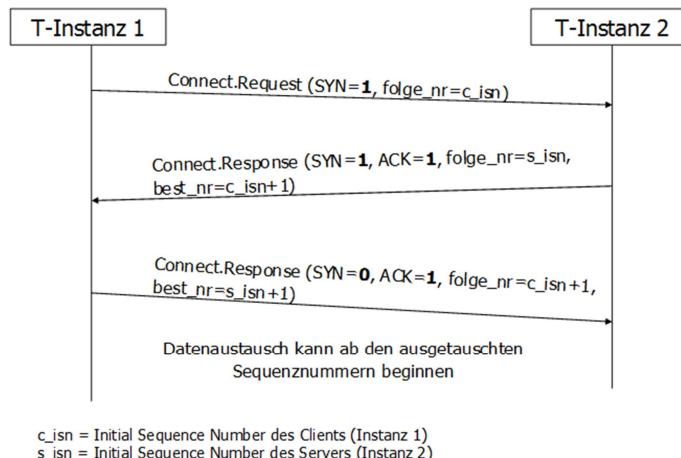


- **Optionen**

- Wurden früher relativ selten verwendet
- Bestehen entweder aus einem einzelnen Byte (Optionsnummer) oder haben eine variable Form
- Optionen sind z.B.:
 - MSS: Maximum Segment Size der Verbindung einstellen
 - SACKOK: Selektive Wiederholungen anstelle von *go back n* einstellen
 - WSOPT (*Windows-Scale-Option*): Maximale Fenstergröße verlängern (2^{30} Byte max. möglich)

Verbindungsaufbau: Parameter aushandeln und Drei-Wege-Handshake

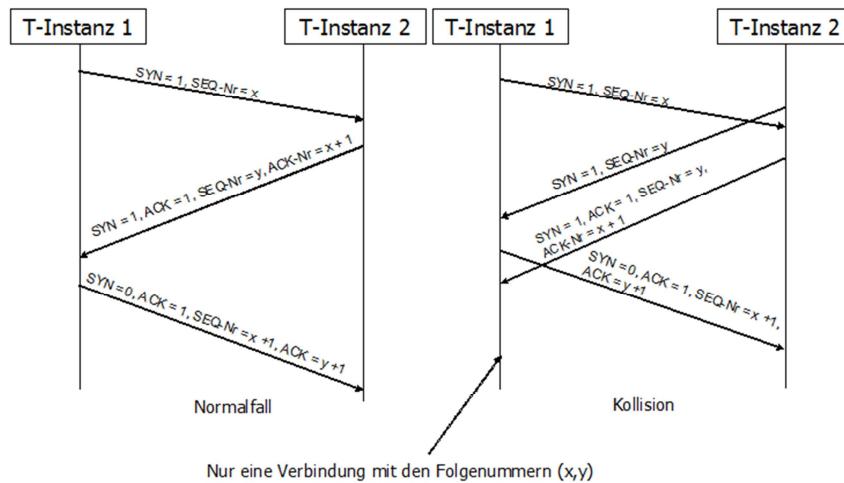
- Beim Verbindungsauftbau werden die **MSS** und die **Sequenznummern** ausgehandelt
- Verwendung des Drei-Wege-Handshake-Mechanismus
 - Initiale Sequenznummern werden berechnet und ausgetauscht
- Kollision beim Verbindungsauftbau ist möglich:
 - Zwei Hosts versuchen gleichzeitig eine Verbindung mit gleichen Parametern zueinander aufzubauen
 - Es wird nur eine TCP-Verbindung aufgebaut
- Normaler Ablauf



Fehlerszenarien

- TCP muss mehrere Fehlersituationen richtig bearbeiten
 - Gleichzeitiger Verbindungsauftbauversuch beider Partner darf nur zu einer Verbindung führen
 - Alte Duplikate von TCP-Segmenten werden beim Verbindungsauftbau empfangen
 - Reset der Verbindung (RST-Bit) und erneuter Aufbau
 - Halb-offene Verbindung erkennen
 - Mit Reset (RST-Bit) abbauen
 - ...

Kollisionsfall



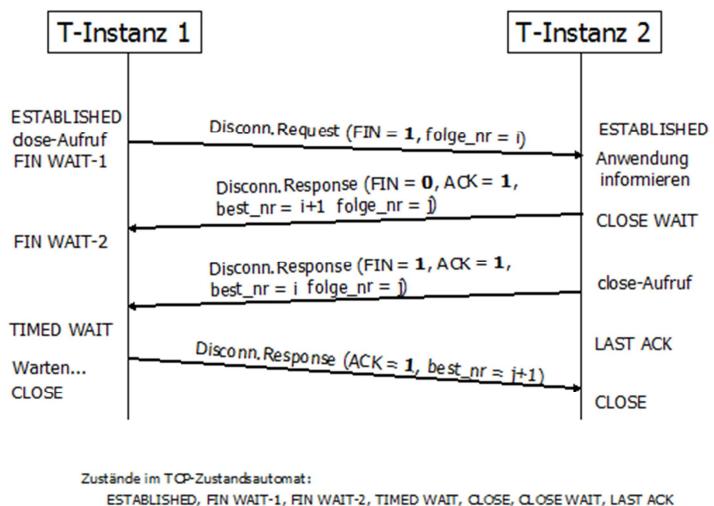
Kollision der Sequenznummern

- **Problem:**
 - Erneuter, schneller Verbindungsauftakt nach Crash könnte zu Sequenznummern-Kollision führen
 - Ein noch altes TCP-Segment könnte bei neuer „Inkarnation“ der Verbindung (gleiche Adressparameter) ankommen und nicht erkannt werden
 - Das muss verhindert werden
- **Lösung zur Synchronisation der Sequenznummern:**
 - Keine netzweit globale Uhr zur Bestimmung der Sequenznummer
 - **Keine feste Vorgabe**, wie ISN ermittelt wird → auch implementierungsabhängig
 - Generierung der ISNs anhand eines (fiktiven) Zeitgebers
 - Vorschlag im RFC 793, S. 28: Zyklus des Zeitgebers von 4,55 Stunden, Zeitgeber wird alle 4 ms erhöht
 - Einsatz des 3-Way-Handshake zum Synchronisieren der Sequenznummern beim Verbindungsauftakt
 - **Max. Segmentlebensdauer (MSL)** berücksichtigen: Wurde im RFC 792 auf 2 Minuten festgelegt
 - Diese Zeit muss gewartet werden, bevor eine neue ISN nach einem Crash einer Verbindung zugewiesen wird
 - Schwierig zu implementieren ohne persistente Speicherung der Verbindungsdaten!

Protokoll

- **Verbindungsabbau-Protokoll:**
 - Modifizierter Dreiwege-Handshake-Mechanismus
 - Jede der beiden Verbindungen der Vollduplex-Verbindung wird abgebaut, d.h. beide Seiten bauen ihre „Senderichtung“ ab
- **Ablauf:**
 - Aktiv abbauender Partner sendet zunächst ein Segment mit **FIN=1**
 - Passiver Partner antwortet zunächst mit einem **ACK** und informiert die Anwendung (Signalisierung implementierungsabhängig!)
 - Wenn die Anwendung **close** aufruft, sendet die Partnerinstanz ebenfalls ein Segment mit **FIN=1**
 - Aktiver Partner sendet abschließend ein Segment mit **ACK=1**

- Client baut die Verbindung ab (auch Server kann es)
- Alle Segmente mit Folgenummer < i bzw. j sind noch zu verarbeiten
- FIN-Segment zählt Sequenznummer **um 1 hoch** (zählt als 1 Datenbyte)



Signalisierung beim passiven Partner

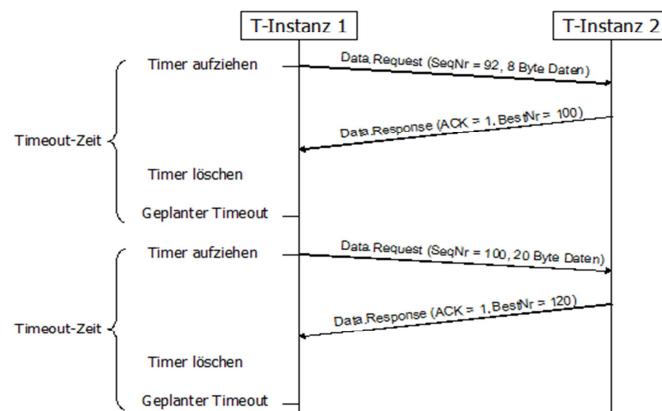
- Laut Zustandsautomat gibt es **mehrere Varianten** des Verbindungsabbaus
 - Es geht auch mit drei Segmenten (FIN=1 + ACK=1)
- Die **Signalisierung** des Verbindungsabbaus an die Anwendung ist der Implementierung überlassen und im RFC nicht genau beschrieben
 - Es kann eine Weile dauern, bis die Anwendung auf das close-Ereignis reagiert
 - Anwendung kann evtl. sogar eine Benutzereingabe erfordern
 - Dies hängt vom Programm ab
- Auch **abnormale Beendigung** einer Verbindung ist möglich
 - Segment mit **RST-Bit=1** wird gesendet und der Empfänger bricht die Verbindung sofort ab

Datenübertragung

- Einsatz von Sequenznummern, die auf einzelnen Bytes, nicht auf TCP-Segmenten basieren (siehe Header->**Sequenznummer**)
- Die Sequenznummer enthält die Nummer des nächsten erwarteten Bytes
- Alle gesendeten Bytes werden vom Empfänger im Feld **Bestätigungsnummer kumulativ** quittiert
- Selektive, positive Quittierung auch möglich: Vorschlag in einem eigenen RFC
- Die Bestätigung muss von der empfangenden TCP-Instanz **nicht unbedingt sofort** gesendet werden, wenn noch Platz im Puffer ist
 - Hier besteht Implementierungsfreiheit für die Hersteller von TCP/IP-Stacks

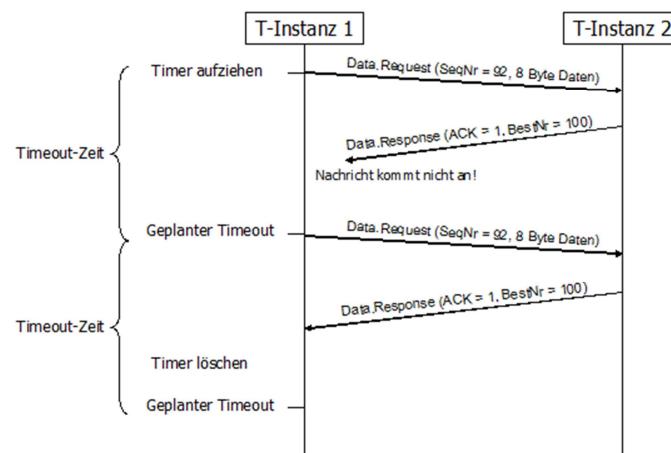
Szenario: Erfolgreiche Übertragung

- Sende-Instanz zieht Timer auf
- Timer wird nach ACK gelöscht



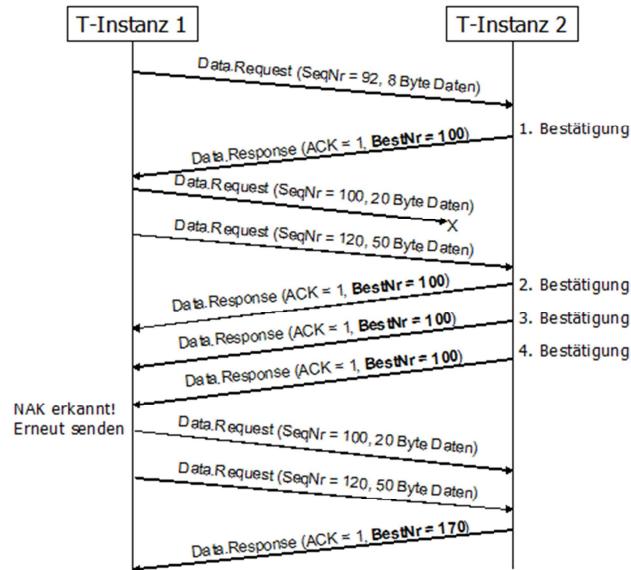
Szenario: Bestätigung geht verloren

- Sende-Instanz zieht Timer auf
- Timer läuft bei verlorengegangener Quittung ab
- TCP-Segment wird erneut gesendet



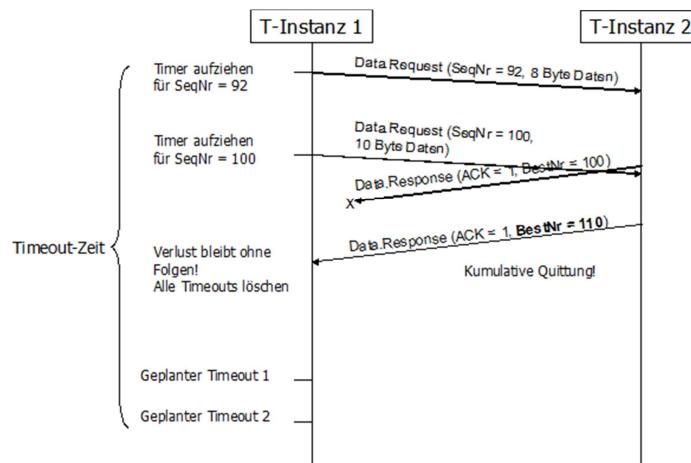
Szenario: NAK bewirkt Sendewiederholung

- Im Beispiel siehe 2. ACK wird **drei mal** vom Empfänger gesendet
- Sender erkennt Problem und sendet erneut → Problem wird vor dem Timerablauf erkannt
- Wird als **implizites NAK** bezeichnet!



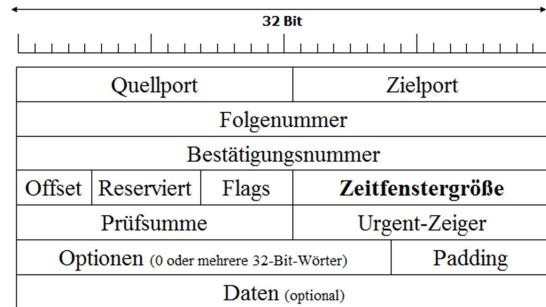
Szenario: Kumulative Quittung verhindert erneutes Senden

- Verlust eines Segments bleibt ohne Folgen

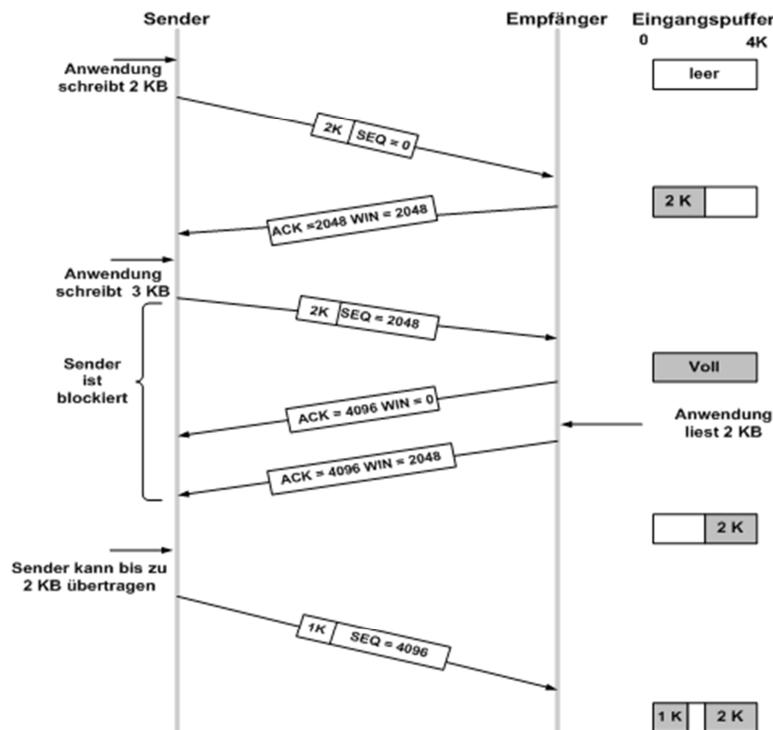


Sliding Window Mechanismus

- Der Sliding Window Mechanismus erlaubt die Übertragung von mehreren TCP-Sequenzen, bevor ein ACKnowlegde eintrifft
- Bei TCP funktioniert Sliding Window auf Basis von Octets (Bytes)
- Die Octets (Bytes) eines Streams sind sequenziell nummeriert
- Flusskontrolle wird über das durch den Empfänger veranlasste Ausbremsen der Übertragung erreicht

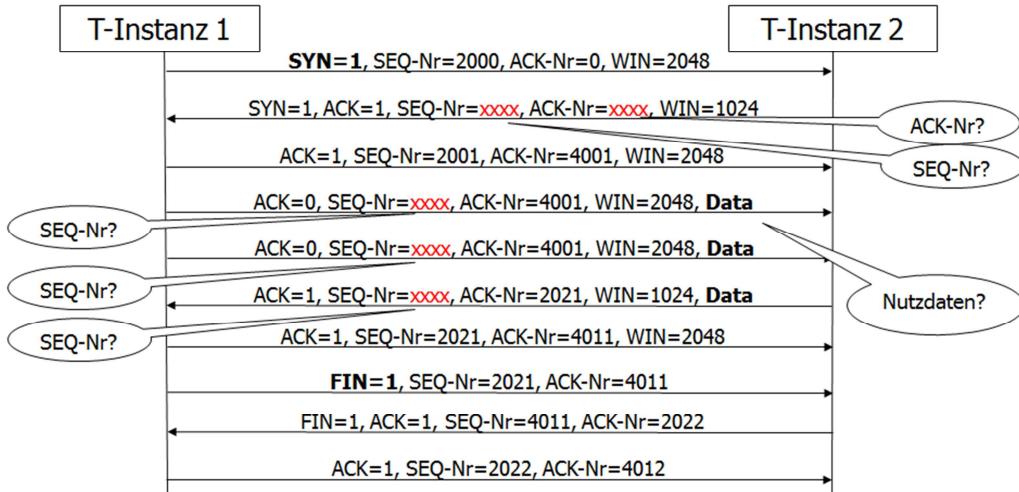


- Die TCP-Instanzen reservieren beim Verbindungsaufbau Puffer für abgehende und ankommende Daten
- Empfänger informiert den Sender über den Füllstand des Empfangspuffers



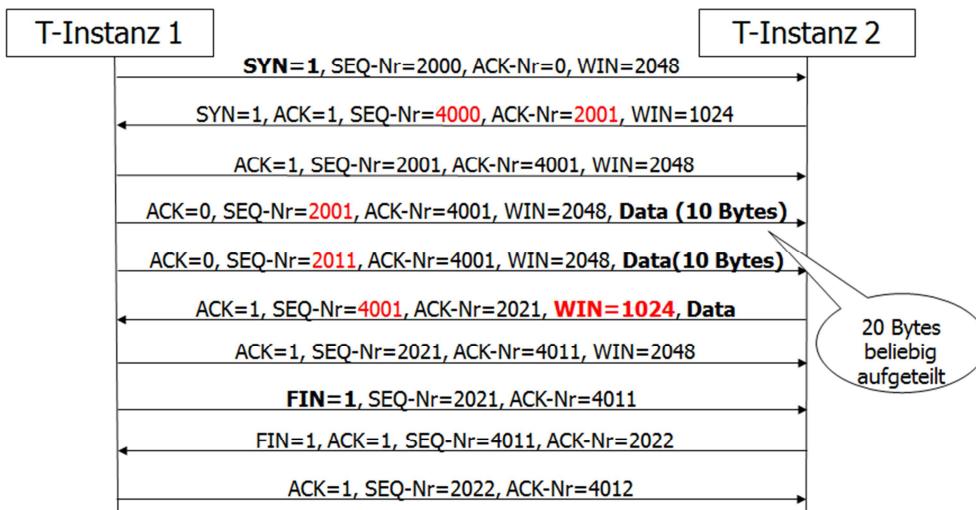
Nachrichtenfluss: Kleine Übung

Ports vernachlässigt



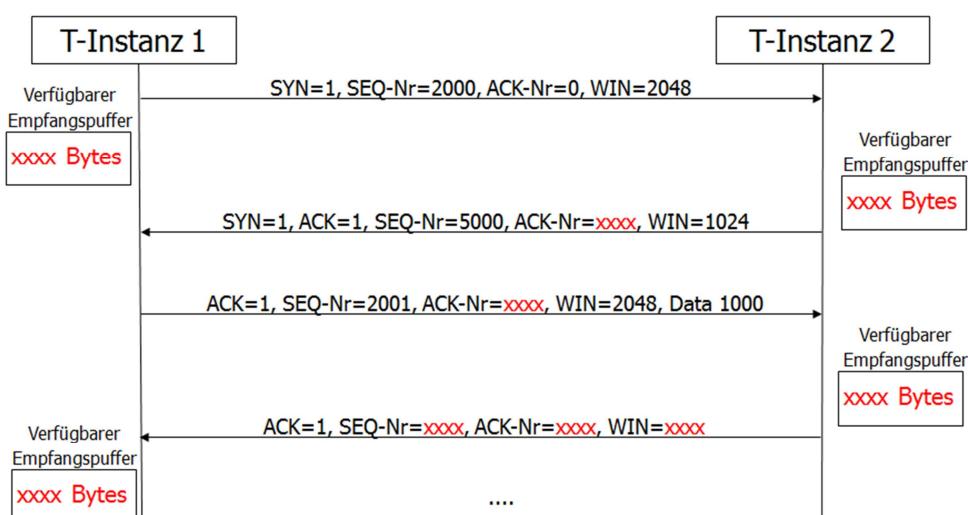
Nachrichtenfluss: Lösung

Ports vernachlässigt

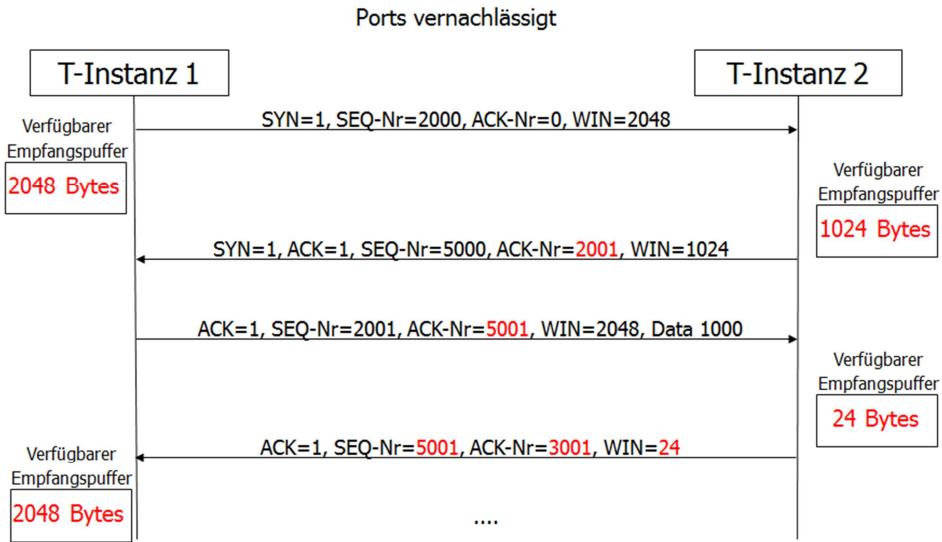


Nachrichtenfluss: Noch eine Übung

Ports vernachlässigt

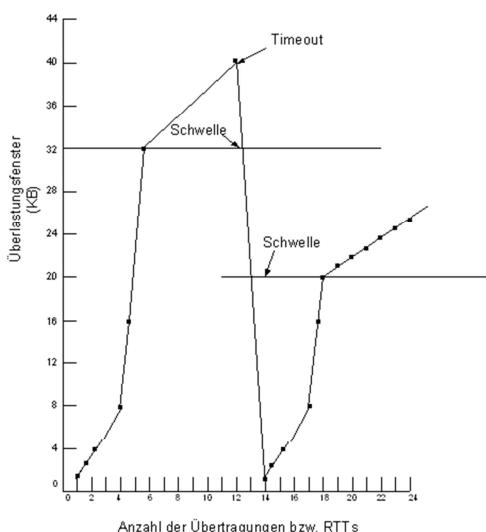


Nachrichtenfluss: Lösung



Staukontrolle bzw. Überlastkontrolle

- **1986** gab es im Internet massive Stausituationen
- Seit **1989** ist Staukontrolle ein wichtiger Bestandteil von TCP
- **IP reagiert nicht** auf Überlastsituationen
- Paketverlust wird von TCP als Auswirkung einer **Stausituation** im Netz interpretiert
- Annahme: Netze sind prinzipiell stabil, **ein fehlendes ACK nach dem Senden einer Nachricht wird als Stau im Netz betrachtet**
- TCP **tastet** sich an die maximale Datenübertragungs-rate einer Verbindung **heran**
 - Dies ist die Größe des Überlastfensters
- Das verwendete Verfahren ist das reaktive **Slow-Start**-Verfahren (RFC 1122)
- Es baut auf das Erkennen von Datenverlusten auf
- Die **Übertragungsrate** wird bei diesem Verfahren im Überlastfall vom Sender massiv **gedrosselt**, die Datenmengen werden kontrolliert
- TCP-Implementierungen **müssen** das Slow-Start-Verfahren unterstützen
 - Erste TCP-Segmentlänge im Beispiel 1 KB (ausgetauscht)
 - 1. Schwellwert bei 32 KB
 - Timeout bei Segmentlänge von 40 KB
 - Schwellwert wird dann auf 20 KB gesetzt



- Es gibt verschiedene Varianten des Slow-Start-Algorithmus
- **Quittungen** dienen als **Taktgeber** für den Sender
- Neben dem Empfangsfenster wird ein neues Fenster eingeführt: das **Staukontrollfenster** (bzw. Überlastungsfenster)
- Es gilt:
Sendekredit für eine TCP-Verbindung =

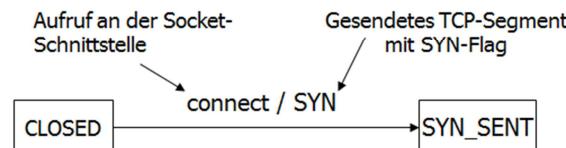
$$\min \{\text{Überlastungsfenster}, \text{Empfangsfenster}\}$$
- **Zwei Phasen:**
 - Slow-Start-Phase
 - Probing-Phase
- Slow-Start-Phase:
 - Sender und Empfänger einigen sich auf eine erste sendbare TCP-Segmentlänge (z.B. 1024 Bytes)
 - Sender sendet Segment dieser Länge
 - Jeweils Verdoppelung der Anzahl an Segmenten bei erfolgreicher Übertragung (exponentielle Steigerung)
 - Ein Schwellenwert (Threshold) wird ermittelt
 - Bei Erreichen des Schwellwerts geht es in die Probing-Phase über

→ Gar nicht so langsam!
- Die Probing-Phase: nicht Klausurrelevant !!!!
- Die Timerlänge ist entscheidend!
- ACK wird nicht empfangen
- Die Probing-Phase: nicht Klausurrelevant !!!!
- Die Timerlänge ist entscheidend!
- Zu lang: Evtl. Leistungsverlust
- Zu kurz: Erhöhte Last durch erneutes Senden
- Dynamische Berechnung anhand der Umlaufzeit eines Segments (vgl. Zitterbart)

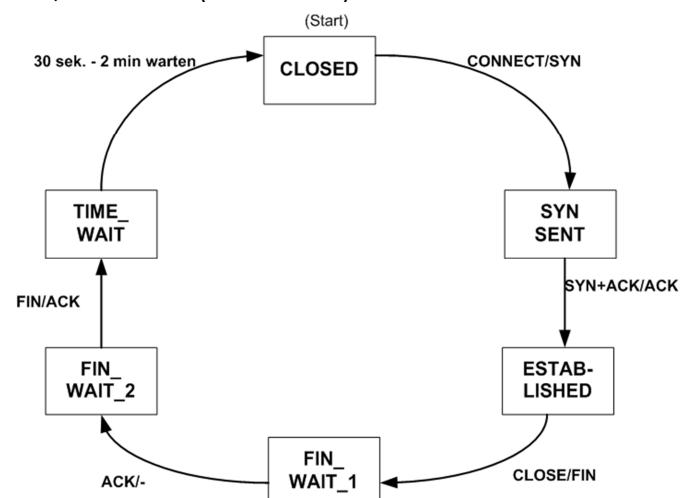
Zustandsautomat

Zustand	Beschreibung
CLOSED	Keine Verbindung aktiv oder anstehend
LISTEN	Der Server wartet auf eine ankommende Verbindung
SYN_RCV	Ankunft einer Verbindungsanfrage und Warten auf Bestätigung
SYN_SENT	Die Anwendung hat begonnen, eine Verbindung zu öffnen
ESTABLISHED	Zustand der normalen Datenübertragung
FIN_WAIT_1	Die Anwendung möchte die Übertragung beenden
FIN_WAIT_2	Die andere Seite ist einverstanden, die Verbindung abzubauen
TIME_WAIT	Warten, bis keine Pakete mehr kommen
CLOSING	Beide Seiten haben versucht, gleichzeitig zu beenden
CLOSE_WAIT	Die Gegenseite hat den Abbau eingeleitet
LAST_ACK	Warten, bis keine Pakete mehr kommen

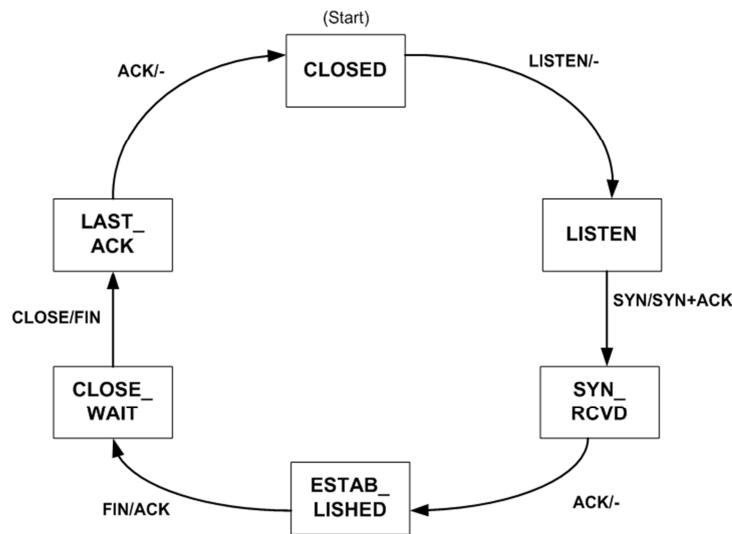
- Ein TCP-Zustandsautomat lässt sich als Quintupel $\langle S, I, O, T, s_0 \rangle$ beschreiben:
 - $S = \{\text{CLOSED}, \text{LISTEN}, \text{SYN_RCV}, \dots\}$
 - $I = \{\text{connect}, \text{send}, \text{close}, \text{SYN}, \text{ACK}, \text{FIN}, \dots\}$
 - $O = \{\text{SYN}, \text{ACK}, \text{FIN}, \dots\}$
 - $s_0 = \text{CLOSED} \in S$
- Hinweis: Wir beschreiben im Weiteren nur die Transitionen des Verbindungsau- und abbaus
- Beispiel einer Transition:
 $\langle S, I, O, T, s_0 \rangle$ beschreiben:
 - $S = \{\text{CLOSED}, \text{LISTEN}, \text{SYN_RCV}, \dots\}$
 - $I = \{\text{connect}, \text{send}, \text{close}, \text{SYN}, \text{ACK}, \text{FIN}, \dots\}$
 - $O = \{\text{SYN}, \text{ACK}, \text{FIN}, \dots\}$
 - $s_0 = \text{CLOSED} \in S$
- Hinweis: Wir beschreiben im Weiteren nur die Transitionen des Verbindungsau- und abbaus
- Beispiel einer Transition:



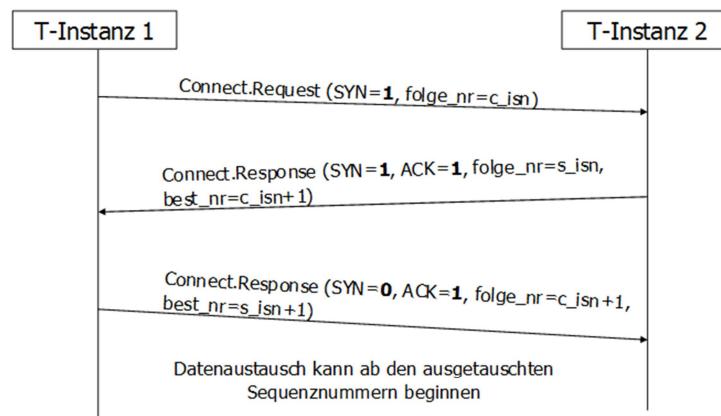
Finite-State-Machine-Modell, TCP-Client (vereinfacht)



Finite-State-Machine-Modell, TCP-Server (vereinfacht)



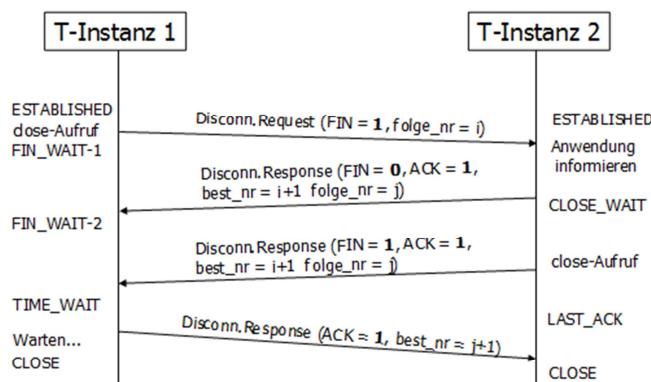
Verbindungsaufbau



c_isn = Initial Sequence Number des Clients (Instanz 1)
 s_isn = Initial Sequence Number des Servers (Instanz 2)

Verbindungsabbau

- Client baut die Verbindung ab (auch Server kann es)
- Alle Segmente mit Folgenummer < i bzw. j sind noch zu verarbeiten

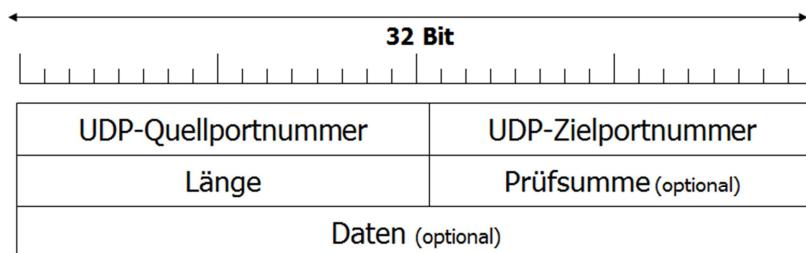


Zustände im TCP-Zustandsautomat:
 ESTABLISHED, FIN_WAIT-1, FIN_WAIT-2, TIME_WAIT, CLOSE, CLOSE_WAIT, LAST_ACK

UDP: Einordnung und Aufgaben

- **Unzuverlässiges**, verbindungsloses Transport-protokoll
 - **Keine Empfangsbestätigung** für Pakete
 - UDP-Nachrichten **können** ohne Kontrolle **verloren gehen**
 - Eingehende Pakete werden **nicht in einer Reihenfolge sortiert**
 - Maßnahmen zur Erhöhung der Zuverlässigkeit müssen im Anwendungsprotokoll ergriffen werden, z.B.
 - ACK und Warten mit Timeout
 - Wiederholtes Senden bei fehlendem ACK
- **Vorteile** von UDP gegenüber TCP
 - Bessere Leistung möglich, aber nur, wenn TCP nicht nachgebaut werden muss
 - Multicast- und Broadcast wird unterstützt
- Bei UDP ist **keine explizite Verbindungsaufbau-Phase** erforderlich und entsprechend auch **kein Verbindungsabbau**
- Userprozess erzeugt ein UDP-Socket und kann Nachrichten senden und empfangen
- Nachrichten werden bei UDP als **Datagramme** bezeichnet
- In den Datagrammen wird die T-SAP-Adresse des Senders und des Empfängers gesendet (UDP-Ports)

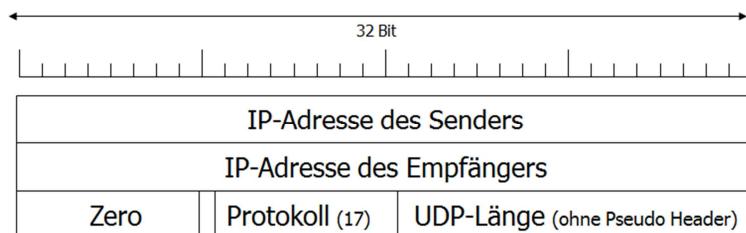
UDP-Header (PCI)



- UDP-Ziel-Portnummer: Nummer des empfangenden Ports
- UDP-Quell-Portnummer: Nummer des sendenden Ports
- Länge: Größe des UDP-Paketes inkl. Header in Byte
- Prüfsumme (optional): Prüft das Gesamtpaket (Daten+Header) mit Pseudoheader (auch so bei TCP)
- Daten: Nettodata der Nachricht

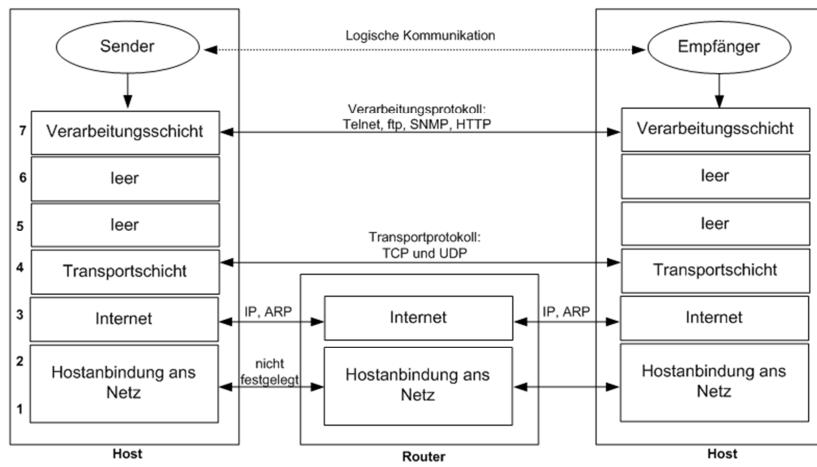
Pseudoheader:

- Die Prüfsumme ist die einzige Möglichkeit, die intakte Übertragung beim Empfänger zu verifizieren
- Vor dem Berechnen der Prüfsumme wird ein Pseudoheader ergänzt und das Paket auf eine durch 16 Bit teilbare Größe aufgefüllt
- Die Prüfsumme wird als Einerkompliment über die gesamte Nachricht (inkl. Pseudoheader) errechnet



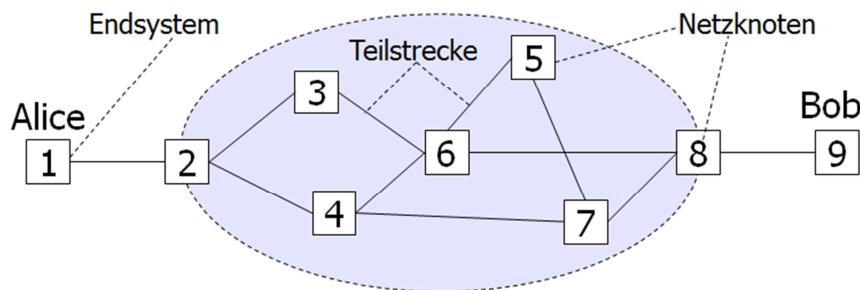
- Der Empfänger muss bei Empfang einer UDP-Nachricht folgendes unternehmen:
 - die IP-Adressen aus dem ankommenden IP-Paket lesen
 - Der Pseudoheader muss zusammengebaut werden
 - Die Prüfsumme muss ebenfalls berechnet werden
 - Die mitgesendete Prüfsumme mit der berechneten vergleichen
- Wenn die beiden Prüfsummen identisch sind, dann muss das Datagramm seinen Zielrechner und auch den richtigen UDP-Port erreicht haben
 - Ziel ist, beim Empfänger herauszufinden, ob das Paket den richtigen Empfänger gefunden hat

TCP/IP-Referenzmodell



Vermittlungsschicht Grundlagen

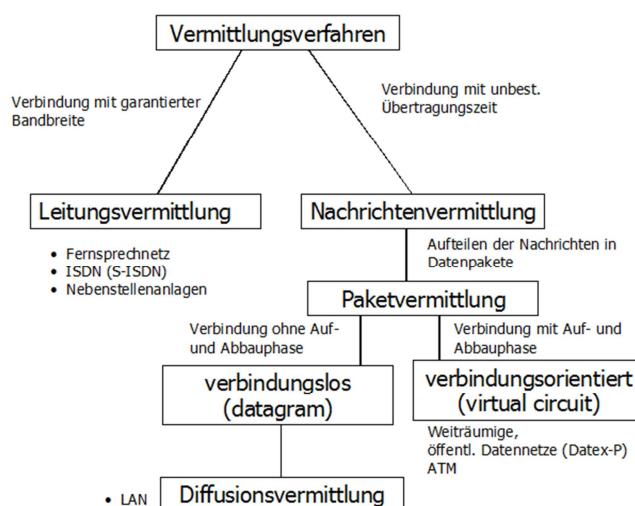
- Zu den Aufgaben gehören:
 - Wegewahl (auch Routing genannt)
 - Multiplexen und Demultiplexen
 - Staukontrolle (Congestion Control)
 - Fragmentierung/Defragmentierung



- **Netzknoten** sind über **Teilstrecken** miteinander verbunden
 - **Endsysteme** sind mit Netzknoten verbunden
 - Netzknoten verwaltet zwei oder mehr Verbindungen
 - Endsystem hat meist nur eine Verbindung
- Die Endsysteme kommunizieren über einen oder mehrere **Netzknotenrechner** (kurz: Netzknoten, Knoten, Router)
- Die Übertragungswege werden von Knoten zu Knoten bereitgestellt (**Teilstrecken**)
- Die **Fehlersicherung** findet auf den Teilstrecken (Schicht zwei) statt
- Die Schnittstelle zur Vermittlungsschicht ist auch meist die **Netzbetreiberschnittstelle**

Den **Gesamtvorgang** der **Verbindungsherstellung**, des **Haltens** und des **Abbauens** einer Verbindung bezeichnet man als **Vermittlung** (engl. Switching).

Welche Vermittlungsverfahren gibt es ?? Zuerst wird unterschieden in....



Leitungsvermittlung:

- Klassisches Switching-Verfahren, auch circuit switching oder Durchschaltevermittlung genannt
- Über die gesamte Verbindung wird ein physikalischer Verbindungsweg durch das Netzwerk geschaltet
- Es wird eine feste Bandbreite garantiert und zwar unabhängig von dem, was tatsächlich übertragen wird
 - Evtl. wird Bandbreite unnötig reserviert
 - Blockierungen (Ablehnung eines Verbindungswunsches), wenn kein Verbindungsweg mehr frei ist
- Beispielnetze, die Leitungsvermittlung nutzen:
 - Analoges Fernsprechnetz
 - Digitales ISDN

(Nachrichtenvermittlung: Bei der Nachrichtenvermittlung werden immer komplette Nachrichten zwischen den Netzwerkknoten ausgetauscht. Nachrichten werden erst weitergesendet, wenn sie vollständig sind. Man spricht hier auch von Store-and-Forward-Verfahren.)

Paketvermittlung:

- Paketvermittlung ist eine Nachrichtenvermittlung:
 - Komplette Nachricht mit Zieladresse wird ins Netz gesendet
 - Netz überträgt die Nachricht evtl. über mehrere Knoten mit Zwischenspeicherung
 - Nachricht wird ggf. in einzelne Pakete (N-PDUs) zerlegt und versendet
- Ist für die Datenübertragung effizienter
- Keine garantierte Bandbreite, dafür aber keine Blockierungen
- Beispiel:
 - Internet Protocol
 - Breitband-ISDN auf Basis von ATM (sehr kurze Pakete, Zellen genannt).

Nutzung von Datagrammen

- Datagramme (N-PDUs) werden bei einer einfachen Paketvermittlung ohne vorhergehenden Verbindungsauflauf verwendet
- Jedes Datagramm enthält die Quell- und die Zieladresse
- Die Knoten **ermitteln** für jedes Datagramm einen **optimalen Weg**
- Wird auch als **verbindungslose** Vermittlung bezeichnet
Nur ein einfacher data-Dienst zum Senden von Datagrammen erforderlich
- Einfache Form der Datagramm-Vermittlung ist die Diffusionsvermittlung
- Hier sendet jeder Knoten die empfangenen Pakete an alle Nachbarknoten weiter
 - mit Ausnahme des sendenden Knotens!
- Sinnvoll bei Netzen mit geringer Knotenanzahl
- Klassische Vermittlungsform in LANs
 - Siehe z.B. Ethernet-LAN

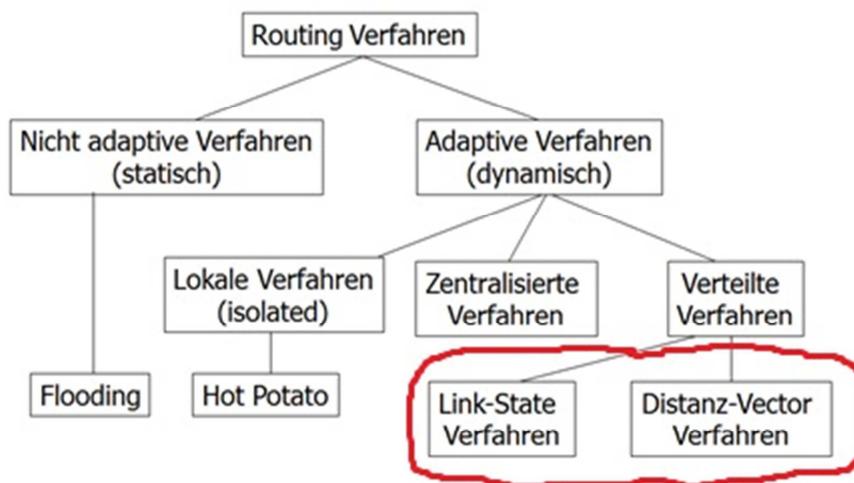
Sonderfälle der Paketvermittlung sind Virtual Circuits.

- Virtual Circuits werden auch „**scheinbare Verbindungen**“ genannt
- Reduzierung des aufwändigen Routens bei jedem Paket durch verbindungsorientiertes Verfahren
- Verbindung bleibt für die Dauer der Datenübertragung erhalten
- Kein physikalisches Durchschalten der Verbindung, sondern Nutzung von Routing-Informationen in den Knoten
- Drei Phasen der Datenübertragung mit entsprechenden Diensten:
 - Verbindungsauflauf (connect-Dienst)

- Datenübertragung (data-Dienst)
- Verbindungsabbau (disconnect-Dienst)
- Die Verbindung zwischen zwei Endsystemen wird schrittweise über Teilstrecken aufgebaut
 - Knoten müssen in der Verbindungsaufbauphase Informationen über das Mapping von eingehenden Paketen zu Ausgangsteilstrecken speichern
 - Statusverwaltung
 - Verbindungstabellen in den Knoten erforderlich

Routing

- Die Wegewahl (Routing) ist eine der wesentlichen Aufgaben der Schicht-3-Instanzen
- Ziel ist es den ‚optimalen‘ Weg zwischen den Endsystemen zu wählen
- Notwendig bei alternativen Wegen zwischen den Endsystemen
- Verschiedene Routing-Kriterien und -Algorithmen sind möglich:
 - Suche der geringsten Entfernung
 - Möglichst geringe Anzahl von Hops (Anzahl der zu durchlaufenden Knoten)
 - Geringste Netzlast
 - ...



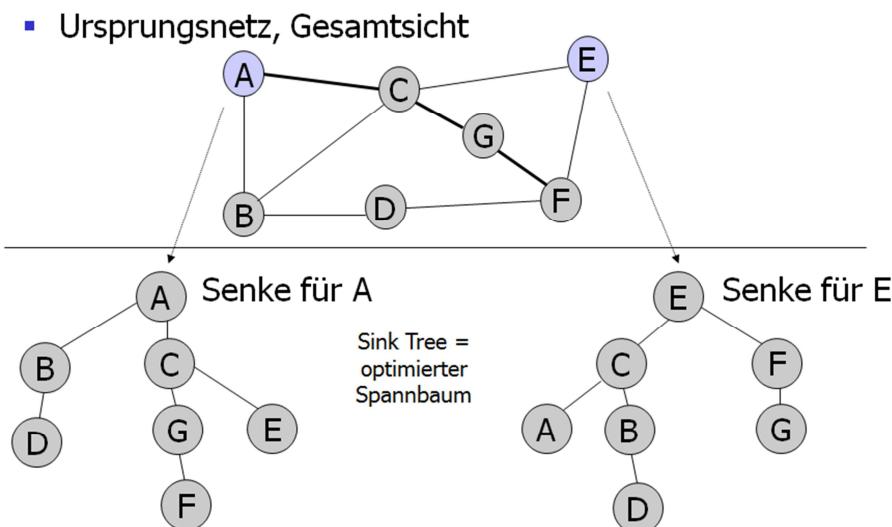
- **Statische** Algorithmen:
 - Keine Messungen, vorher ermittelte Metriken
 - Statische Routing-Tabellen, die bei der Knotenkonfiguration eingerichtet werden (vor Beginn des Betriebs)
- **Dynamische** (adaptive) Algorithmen:
 - Verkehrsmessungen
 - Routing-Tabellen werden dynamisch angepasst (Metriken)
 - Optimierungskriterien können sich dynamisch verändern und werden im Algorithmus berücksichtigt
 - Möglichkeiten:
 - **Isoliertes Routing:** Knoten trifft Entscheidungen alleine
 - **Zentrales** Routing über einen zentralen Knoten (Routing-Kontroll-Zentrum), Zentrale ermittelt und überträgt alle Routing-Tabellen
 - **Dezentrales** Routing mit Routing-Funktionalität in jedem Knoten

- **Statische** Algorithmen:
 - Shortest-Path-Routing
 - Flooding
- **Dynamische** (adaptive) Algorithmen (heute üblich in modernen Netzen):
 - Distance-Vector-Routing
 - ursprünglicher Algorithmus im ARPANET, RIP
 - Link-State-Routing
 - löste Distance-Vector-Routing Ende der 70er im ARPANET ab
 - Hierarchisches Routing

Verteiltes Routing: Beim verteilten Routing unterscheidet man im Wesentlichen zwei verschiedene Verfahren. Zum einen ist das Distance-Vector-Routing oder Entfernungsvektorenverfahren, ein sehr bekanntes und ursprünglich im ARPANET und auch nachher im Internet eingesetztes Verfahren. Zum anderen ist das heute ebenfalls im Internet eingesetzte Verfahren namens Link-State-Routing oder Verbindungszustandsverfahren sehr bekannt. Es wird seit dem Ende der 70er Jahre im ARPANET bzw. im Internet eingesetzt.

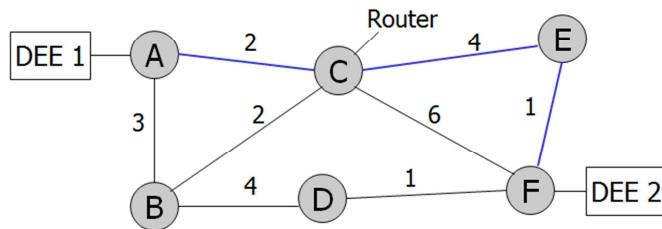
Routing – Optimalitätsprinzip

- Das Optimierungsprinzip (Optimalitätsprinzip nach Richard Bellmann) besagt:
 - Wenn Router C auf dem optimalen Pfad zwischen A und F liegt, dann fällt der Pfad von C nach F ebenso auf diese Route
 -
- Optimalitätsprinzip angewendet auf das Routing:
 - Die optimalen Routen von allen Quellen zu einem bestimmten Ziel bilden einen **Baum**, dessen Wurzel das Ziel ist
 - Dieser Baum enthält keine Schleifen und heißt **Sink Tree** oder **Senke** (optimierter Spanning Tree)
- Ziel von Routing-Algorithmen
 - Senken für alle Router ermitteln
 - Senken für das Routing nutzen



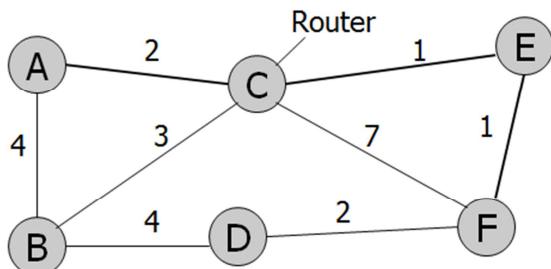
Beispiel - Shortest-Path-Routing

- Graph des Teilnetzes wird erstellt (statisch)
 - Knoten entspricht Router
 - Kante entspricht einer Leitung zwischen zwei Routern
- Kante wird beschriftet („Pfadlänge“), die Metrik hierfür kann berechnet werden aus
 - Entfernung
 - Bandbreite
 - Durchschnittsverkehr
 - Durchschnittliche Warteschlangenlänge in den Routern
 - Verzögerung
 - ...
- Berechnung des kürzesten Pfads z.B. über Dijkstras oder Bellmanns Algorithmus
 - Beispiel: Der kürzeste Pfad zwischen DEE 1 und DEE 2 geht von A nach F über A C E F
 - Summierte Pfadlänge = 7



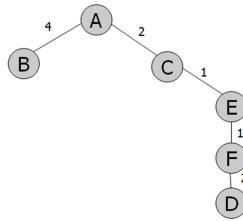
Routing – Übung

- Ermitteln Sie die Summe der Kantengewichte zu den Sink Trees (optimale Spannbäume) des vorliegenden kantengewichteten Graphen für die Knoten A und E und zeichnen Sie die Graphen.
- Beschriften Sie die Kanten mit!



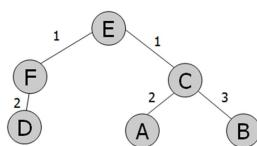
Routing – Lösung zur Übung (1)

- Sink Tree (= optimaler Spannbaum) für A
- Summe der Kantengewichte = 10



Routing – Lösung zur Übung (2)

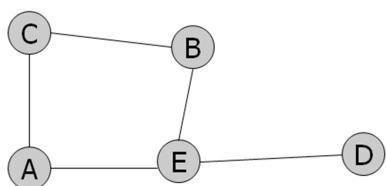
- Sink Tree (= optimaler Spannbaum) für E
- Summe der Kantengewichte = 9



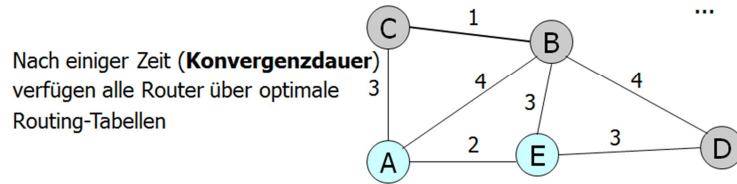
Beispiel – Distance-Vector-Routing

- Andere Bezeichnung: Bellman-Ford-Routing (Bellman, 1957 und Ford, 1962)
- Jeder Router führt eine **dynamisch aktualisierte Routing-Tabelle** mit allen Zielen
 - Einträge enthalten bevorzugte Ausgangsleitung zu einem Ziel
- **Metrik** kann z.B. sein:
 - Verzögerung in ms
 - Anzahl der Teilstrecken (**Hops**) zum Ziel
- Verteilt – iterativ – asynchron (unabhängig voneinander)
- **Benachbarte** Router tauschen Routing-Information aus
- **Schleifen** möglich „**Count-to-Infinity-Problem**“
 - Bei Ausfall eines Links evtl. keine Terminierung mehr sichergestellt
- **Gute Nachrichten** verbreiten sich schnell
- **Schlechte Konvergenz**
 - Schlechte Nachrichten verbreiten sich sehr langsam in Netz
- Routing-Informationen werden nur zwischen Nachbarn ausgetauscht
- Es wird nur die Sicht der Nachbarn und nicht die gesamte Topologie kommuniziert
- Beispiel:
 - C kommuniziert nur mit A und B
 - E Kommuniziert nur mit A, B und D
 - D kommuniziert nur mit E,...
 - B teilt z. B. C mit, dass er E über einen und D über zwei Hops erreichen kann

Ziel	Distanz	Nächster Knoten	Hops



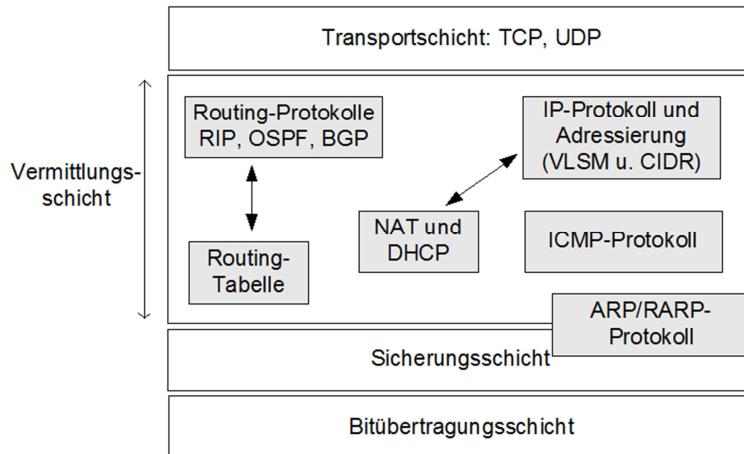
Knoten A				Knoten E			
Ziel	Distanz	Nächster Knoten	Hops	Ziel	Distanz	Nächster Knoten	Hops
B	4	B	1	A	2	A	1
C	3	C	1	B	3	B	1
E	2	E	1	C	4	B	2
D	5	E	2	D	3	D	1



Link-State-Routing

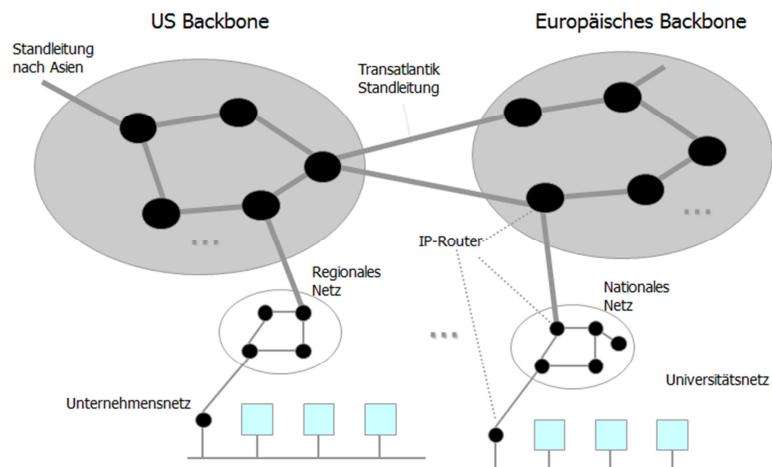
- Jeder Router verwaltet eine **Kopie der Netzwerktopologie** (Link-State-Datenbank)
- **Zielsetzung:** Jeder Knoten muss alle Kosteninformationen kennen
- Jeder Router verteilt die lokale Information per Flooding an alle anderen Router im Netz
- Die Berechnung der Routen erfolgt **dezentral**
- Jeder Knoten errechnet den **absolut kürzesten Pfad**
- **Berechnung der kürzesten Pfade** z.B. über Shortest-Path-Algorithmus (z.B. Dijkstras Algorithmus)
- **Keine Schleifen** möglich, da jeder Knoten die gleiche Information über die Topologie besitzt
- Schnelle Reaktion auf Topologieänderungen möglich

Vermittlungsschicht im Internet



- Das Internet ist eine **hierarchische Organisation** des Netzwerks
- Große Backbones** sind über Leitungen mit hoher Bandbreite und schnellen Routern verbunden
- An den Backbones hängen **regionale Netze**
- An den regionalen Netzen hängen die **Netze von Unternehmen, Universitäten, Internet Service Providern (ISP),...**
- Der Zugriff über das Internet von Deutschland aus auf einen Server in den USA wird über mehrere IP-Router geroutet

Überblick über das Internet, Backbone

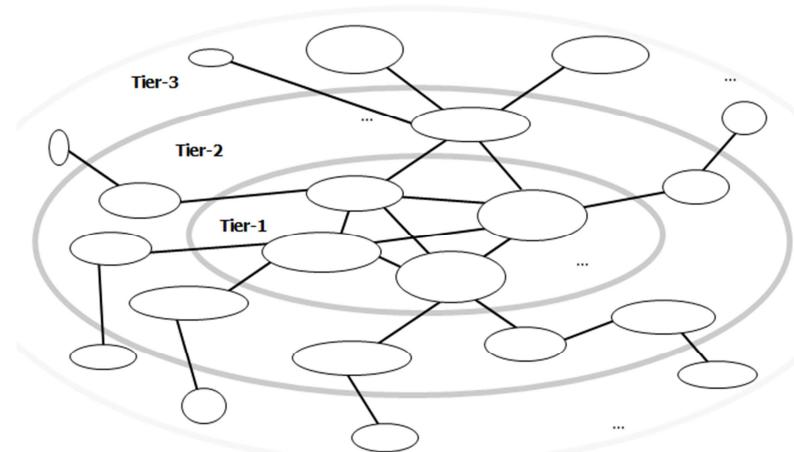


- Autonome Systeme haben sich unterschiedlich entwickelt → z.B. verschiedene Routing-Strategien
- Es gibt derzeit mehr als 110.000 autonome Systeme weltweit
- Jedes AS hat eine **eindeutige Nummer**
 - 11, Harward-University
 - 1248, Nokia
 - 2022, Siemens
 - 3680, Novell
 - 4183, Compuserve
 - 6142, Sun
 - 12816, MWN

Begriffe:

- Autonome Systeme = AS
- Internet-Provider = Provider
 - Meist auch Synonym für Provider
- Tier-1-AS, Tier-2-AS, Tier-3-AS
 - Edge-Networks = Tier-1-AS
- Internet-Knoten
 - = IX (Internet Exchange) = IXP (Internet Exchange Point)
 - = NAP (Network Access Point) → USA
- Public und private Peering-Points
- Peering-Agreement = Peering-Vereinbarung:
 - Übertragung ohne Gebühr bei gleichberechtigten Peers
- Transit-Vereinbarung:
 - Gebühr nach vereinbarter Bandbreite

Das Internet heute, Verbindungen zwischen AS



- **Tier-3:** Kleine, lokale Provider, Endkundengeschäft
 - M-net in Bayern (Hauptgesellschafter: Münchener Stadtwerke)
 - Hansenet (Hamburg, Tochter der Telecom Italia)
 - Versatel (Berlin)
- **Tier-2:** Betreiber großer, überregionaler Netze
 - Deutsche Telekom
 - France Telecom und France Telecom
 - Tiscali (Telekom-Unternehmen, Italien)
- **Tier-1:** Betreiber von globalen Internet-Backbones
 - AT&T (US-amerikanischer Telekomanbieter)
 - AOL (US-amerikanischer Online-Dienst)
 - NTT (Nippon Telegraph and Telephone Corporation)
 - Verizon Communications (US-amerikanischer Telekomanbieter)
- Zuständig für die Weiterentwicklung des Internet ist das IAB (Internet Activity (heute: Architecture) Board), das bereits **1983** gegründet und **1989** umorganisiert wurde.
- Das **IAB** (Board) bestimmt die Richtlinien der Politik
- Die **IETF** kümmert sich in verschiedenen Bereichen (areas) um kurz- und mittelfristige Probleme
- Die **IESG** koordiniert die IETF Working Groups
- Die **IRTF** ist ein Forschungsbereich, der die TCP/IP-Forschungsthemen koordiniert

- Die **IRSG** koordiniert die Forschungsaktivitäten der einzelnen Gruppen
- Die Working Groups setzen sich aus freiwilligen Mitarbeitern zusammen
- Das NIC (Network Information Center, gesprochen NICK), ist zuständig für
 - die Dokumentation und
 - die Verwaltung der umfangreichen Information über
 - Protokolle,
 - Standards,
 - Services, usw.
- Das NIC verwaltet das Internet, z.B. auch die Domänennamen (www.nic.net)
- In Deutschland ist die DENIC (www.denic.de) als nationale Vertretung eingerichtet (Frankfurt)
- Die Dokumentation und die Standards werden als technische Reports gesammelt und heißen RFCs (Request for Comments)
- RFCs durchlaufen während ihrer Lebenszeit verschiedene Stadien (Proposed Standard → Draft Standard → Internet Standard)
- Manche RFCs werden auch nie zum Internet Standard
- RFCs sind frei verfügbar (z.B. unter www.rfc-editor.org)

Internet Protocol:Hauptaufgaben

- **Paketvermitteltes** (datagramm-orientiertes) und verbindungsloses Protokoll der Vermittlungsschicht
- IP dient der **Beförderung von Datagrammen** von einer Quelle zu einem Ziel evtl. über verschiedene Zwischenknoten (IP-Router)
- **Routing-Unterstützung** ist eine zentrale Aufgabe von IP (Routingprotokoll basiert auf diversen Protokollen)
- Datagramme werden während des Transports zerlegt und am Ziel wieder zusammengeführt, bevor sie der Schicht 4 übergeben werden = **Fragmentierung**
- IP stellt einen **ungesicherten verbindungslosen** Dienst zur Verfügung
 - Es existiert **keine Garantie der Paketauslieferung**
 - Die Übertragung erfolgt nach dem **Best-Effort-Prinzip**
 - „Auslieferung nach bestem Bemühen“
 - Jedes Paket des Datenstroms wird **isoliert** behandelt
 - Das IP-Paket wird in einem Rahmen der zugrundeliegenden Schicht 2 transportiert, für den Längenrestriktionen bestehen:
 - bei Ethernet ist eine Länge von 1500 Bytes üblich
 - MTU = Maximum Transfer Unit

IPv4 Adressierung im Internet, IP-Adressen

- IP-Adressen sind **32 Bit** lange Adressen
 - **Tupel** aus (Netzwerknummer, Hostnummer)
- IP-Adresse von Quelle und Ziel werden **in allen** IP-Paketen mit übertragen
- Es gibt verschiedene Adressformate
 - man unterscheidet die Klassen A, B, C, D und E
- Schreibweise für IP-Adressen in gepunkteten Dezimalzahlen, jeweils ein Byte als Dezimalzahl zwischen 0 und 255
 - Beispiel:
 - Hexformat: 0xC0290614
 - Dezimalzahl: 192.41.6.20

Addressierung im Internet, IP-Adressformate

4 Byte, 32 Bit			
Klasse A	0	Netz	Host
Klasse B	10	Netz	Host
Klasse C	110	Netz	Host
Klasse D	1110	Multicast-Adresse	
Klasse E	11110	Reservierte Adresse	

Alle Adressen der Form 127.xx.yz sind Loopback-Adressen!

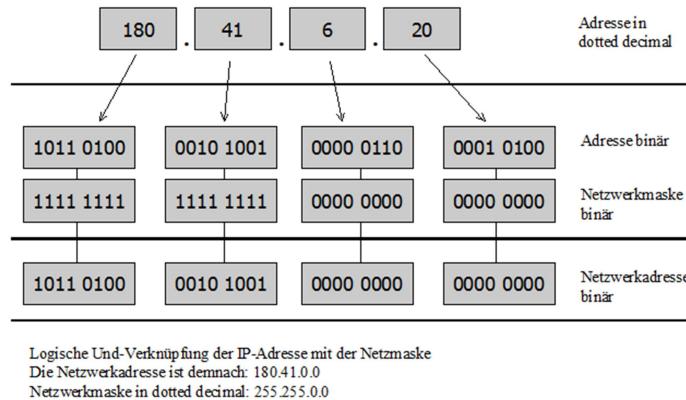
- Die niedrigste IP-Adresse ist **0.0.0.0**
- Die höchste IP-Adresse ist **255.255.255.255**
- Die Adresse **0.0.0.0** hat die besondere Bedeutung „ein bestimmtes Netz“ oder „ein bestimmter Host“
- Die Adresse **255.255.255.255** (-1) wird als Broadcast-Adresse verwendet
 - Limited Broadcast → im lokalen Netz
- Weiterer Broadcast-Typ:
 - Directed Broadcast → Broadcast an anderes Netzwerk
- Netznummern werden vom NIC bzw. in Deutschland von der DENIC zugewiesen!
- **Adressenknappheit** wird erwartet:
 - Neue Version **IPv6** soll hier Abhilfe schaffen

Subnetting

- Die Hostadresse kann zu besseren organisatorischen Gliederung noch mal zur **Subnetz**-Bildung in zwei Teile zerlegt werden:
 - Teilnetznummer
 - Hostnummer
 - Außerhalb des Teilnetzes ist die Aufgliederung nicht sichtbar
- IP-Router in einem Teilnetz berücksichtigen die Subnetzadresse
- **Netzwerkmaske** wird als Bitmaske verwendet, um die Bits der Subnetzwerknummer zu identifizieren
- Beispiel mit Klasse B
- Der lokale Administrator ~~besitzt alle Freiheiten zur~~ Bildung von Subnetzen, ohne die Komplexität auf den Internet-Router zu übertragen
- Bei einer Klasse B-Adresse wäre folgende Struktur denkbar:
 - 3. Byte gibt die Organisationseinheit im Unternehmen an (Subnetz-Adresse)
 - 4. Byte erlaubt die Nummerierung der Geräte: (Stationsadresse)
 - Netzkomponenten (Switch, Hub, etc.): 1 - 9
 - Arbeitsplätze: 10 - 249
 - Server: 250 - 254

Beispiel

- Hier handelt es sich um eine Klasse B-Adresse, warum?



- Problem:
 - Vergeudung von vielen IP-Adressen durch die Aufteilung des Adressraums in Klassen (siehe Klasse-B-Adressen)
 - Aber die Routing-Tabellen der Router sind schon sehr groß und sollten nicht weiter wachsen
- VLSM und CIDR helfen, die Adressproblematik etwas abzumildern
- **VLSM** = Variable Length Subnet Mask
- **CIDR** = Classless InterDomain Routing
- CIDR ist VLSM im öffentlichen Internet!
- Konzept von VLSM/CIDR:
 - Restliche Klasse-C-Netze (ca. 2 Millionen) werden in Blöcken variabler Länge vergeben → Vergabe durch ISP
 - Beispiel: Braucht ein Standort 2000 Adressen erhält er acht aufeinanderfolgende Klasse-C-Netze zugewiesen, kann also auf eine B-Adresse verzichten
- Weitere Verbesserung für das Routing durch Zuordnung der Klasse-C-Adressenbereiche zu Zonen, z.B.
 - Europa: 194.0.0.0 bis 195.255.255.255
 - Nordamerika: 198.0.0.0 bis 199.255.255.255
 - Europäischer Router erkennt anhand der Zieladresse, ob ein Paket in Europa bleibt oder z.B. zu einem amerikanischen Router weitergeleitet werden soll

CIDR und VLSM: Netzwerkpräfix-Notation

- Netzwerkpräfix-Notation (NP-Notation) ermöglicht die Angabe der Netz-Id-Bits in der IP-Adresse
- Notationsbeispiel: 194.24.19.25/20
 - IP-Adresse binär:
 - **11000010.00011000.00010011.00011001** -> Klasse C
 - Die Präfixlänge (hier 20) gibt die Anzahl der fortlaufenden Einsen in der Netzmase an:
 - Netzmase der IP-Adresse:
 - **11111111.11111111.11110000.00000000** = 255.255.240.000

- Klasse A = /8
- Klasse B = /16
- Klasse C = /24

Beispiel

- Cambridge University benötigt 2000 (fast 2^{11}) öffentliche IP-Adressen
 - Klasse-C-Netz mit max. 256 (2^8) Adressen reicht nicht aus
 - Alternative ist ein Klasse-B Netz mit 65536 (2^{16}) Adressen

-> 63488 öffentliche IP-Adressen werden nicht benötigt und somit verschwendet! (über 95% der bereitgestellten Adressen!!!)

- Besser: Nutzung mehrerer zusammenhängender Klasse-C Netze
 - Für den Hostanteil der Adresse werden zusätzlich 3 Bit benötigt (2^{11} Adressen)
 - Cambridge University wird folgender Adressbereich zugewiesen:

194.24.0.0/21 -> 194.24.0.0 bis 194.24.7.255 (Netzwerkmaske 255.255.248.0)

oder

11000010.00011000.00000000.00000000 bis

11000010.00011000.00000111.11111111 mit

11111111.11111111.11111000.00000000 als Netzwerkmaske

Beispiel

- Nach dem gleichen Verfahren werden auch den Universitäten Oxford und Edinburgh mehrere Klasse-C Netze zugewiesen
 - Oxford benötigt 4000 (fast 2^{12}) öffentliche IP-Adressen
 - Edinburgh benötigt 1000 (fast 2^{10}) öffentliche IP-Adressen
- Folgende Adressbereiche werden zugewiesen:
 - Cambridge: 194.24.0.0/21 194.24.0.0 bis 194.24.7.255
 - Oxford: 194.24.16.0/20 194.24.16.0 bis 194.24.31.255
 - Edinburgh: 194.24.8.0/22 194.24.8.0 bis 194.24.11.255



- Ein Standard IP-Router kann die mittels VLSM zusammengefassten Klasse-C-Netze nicht erkennen
- Das Routing muss um CIDR erweitert werden

Routingtabelle ohne CIDR

194.24.0.0	-> Cambridge
194.24.1.0	-> Cambridge
...	
194.24.7.255	-> Cambridge
194.24.8.0	-> Edinburgh
...	
194.24.11.255	-> Edinburgh
194.24.16.0	-> Oxford
...	
194.24.31.255	-> Oxford

Routingtabelle mit CIDR

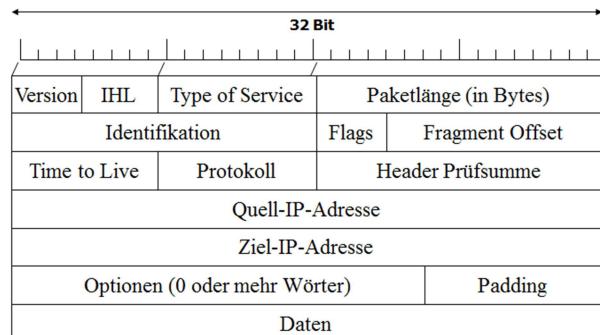
194.24.0.0/21	-> Cambridge
194.24.8.0/22	-> Edinburgh
194.24.16.0/20	-> Oxford

Nur 3 statt 28 Einträge!!!

Übung

- Welche IP-Adressen repräsentiert die CIDR-Adresse 180.41.214.192/28 (Netzwerkpräfix-Notation)?
- Lösung:
 - Insgesamt 4 Bit für Hosts
 - Adressbereich von 180.41.214.192 bis 180.41.214.207
 - Also von 1011 0100 . 0010 1001 . 1101 0110 . 1100 **0000**
 - bis 1011 0100 . 0010 1001 . 1101 0110 . 1100 **1111**
 - Das sind 16 IP-Adressen, 14 davon sind nutzbar

IPv4 Protokollheader



- **Version:** Spezifiziert die genutzte IP-Version; z.Zt. Wechsel von IPv4 auf IP Next Generation (IPv6)
- **IHL:** Gibt die Länge des Paket-Headers an, gemessen in 32-Bit-Worten; ist aufgrund der variablen Länge des Optionsfeldes nötig (mind. 5 → keine Option, max. 15 Worte → 60 Byte)
- **Type of Service:** Dieses 8-Bit-Feld ist wiederum aufgeteilt in:
 - Priorität (3 Bit): 000=Standard, 001=Priorität, ..., 101=kritisch, ...
 - ToS-Spezifikation (3 Bit): Flags zur Angabe von Servicetypen (hoher Durchsatz, niedrige Verzögerung, ...)
 - 2 unbenutzte Bit, **IPv4 nutzt Type of Service nicht**
- **Paketlänge:** Gesamtlänge des Datenpaketes inkl. Header; gemessen in 8-Bit-Worten
 - max. 65.535 Byte
- **Identifikation:** Alle Fragmente eines Datagramms erhalten hier den gleichen Wert
- **Flags:** (3 Bit) Dient der Kontrolle der Fragmentierung
 - Geben an, ob das Feld geteilt werden muss und weitere Pakete folgen oder ob das aktuelle Paket das letzte ist
 - Drei Flags, erstes unbenutzt: 0|DF|MF
 - DF=1 → Fragmentierung ist nicht erlaubt
 - More Fragments=0 → letztes Fragment; MF=1 weitere folgen
- **Fragment Offset:** Dient der korrekten Herstellung der Ursprungssequenz, da Pakete das Ziel in unterschiedlicher Reihenfolge erreichen können, gemessen in 8-Byte-Worten
 - Dient der Ermittlung der relativen Lage des Fragments im Datagram
 - 13 Bit
- **Time to live (TTL):** Gibt an, wie lange ein Datagramm im Internet verbleiben darf
 - Es dient dazu, zu alte Pakete vom Netz zu nehmen, bei 0 wird Paket verworfen und eine ICMP-Nachricht zum Quellhost gesendet
 - War gedacht als Zeit in Sekunden (max. 255 s)
 - Wird aber als Hop-Count genutzt, jeder Router subtrahiert 1
- **Protokoll:** Definiert das darüberliegende Protokoll, an das IP die Daten des Pakets weiterreicht (6=TCP, 89=OSPF,...)

- **Header-Prüfsumme:** Header-Absicherung
 - 16-Bit-Wörter aufsummieren und Einerkomplement bilden
 - Prüft also **nur** den Header, Daten in höheren Protokollen prüfen!
 - Wird für jede Teilstrecke neu berechnet werden, warum?
- **Quell-IP-Adresse und Ziel-IP-Adresse:**
 - Jeweils 32 Bits
 - Identifikation der einzelnen Endsysteme (Hosts)
 - Hier stehen die IP-Adressen von Sender- und Empfängerhost
- **Optionen:** Zusätzliche, optionale Angaben:
 - *Loose Source Routing* → Möglichkeit, den Weg eines Paketes durch das Internet partiell vorgeben; RFC 791
 - *Strict Source Routing* → die Pakete müssen die Pfadvorgabe einhalten (max. 9 Knoten in der Route), RFC 791
 - *Record Routing* → Jeder durchlaufene Router trägt seine Adresse ein, ...
 - Wird selten verwendet!!
- **Padding:** Wenn eine Option genutzt wird, ist das Datagramm bis zur nächsten 32-Bit-Grenze mit Nullen aufzufüllen
- **Daten:** Die Nutzdaten der höheren Schicht

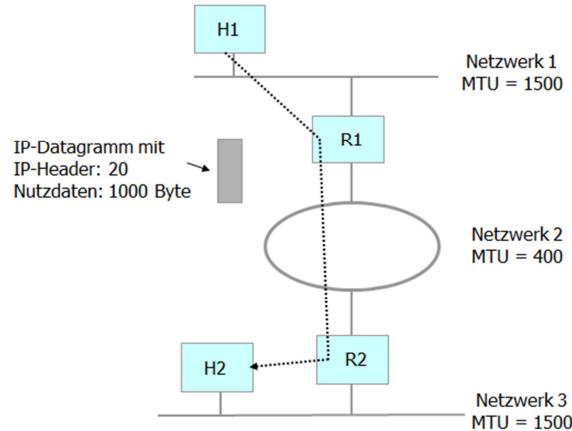
Fragmentierung

Ablauf

- Wenn ein IP-Paket von einem Netzknoten zum anderen weitergeleitet wird, muss es
 - evtl. verschiedene physikalische Netze durchqueren
 - in unterschiedlich zulässige Paketgrößen aufgeteilt werden
- Daher besteht die Notwendigkeit, IP-Datagramme zu zerlegen und am Ziel wieder zusammenzusetzen
 - Fragmentierung und Defragmentierung
 - Alle Router müssen Fragmente der Größe **576 Byte** oder kleiner akzeptieren
- Sobald eine Fragmentierung einsetzt, laufen in einem Knoten mehrere Schritte ab
- Das **DF-Flag** wird überprüft, um festzustellen, ob eine Fragmentierung erlaubt ist. Ist das Bit auf „1“ gesetzt, wird das Paket verworfen
- Ansonsten wird entspr. der zulässigen Paketgröße das Datenfeld des Ur-Paketes in mehrere Teil zerlegt
- Alle neu entstandenen Pakete weisen - mit Ausnahme des letzten Paketes - eine Länge mit einem **Vielfachen von 8 Byte** auf
- Alle Datenteile werden in neu erzeugte IP-Pakete eingebettet. Die Header dieser Pakete sind Kopien des Ursprungskopfes mit einigen Modifikationen
- Header-Modifikation bei der Fragmentierung:
 - Das **MF-Flag** wird in allen Fragmenten mit Ausnahme des letzten auf „1“ gesetzt
 - Das **Fragment-Offset**-Feld enthält Angaben darüber, wo das Datenfeld in Relation zum Beginn des nicht fragmentierten Ur-Paketes platziert ist
 - Enthält das Ur-Paket Optionen, wird abhängig vom Type-Byte entschieden, ob die Option in jedes Paketfragment aufgenommen wird (z.B. Protokollierung der Route)
 - Die **Headerlänge (IHL)** und die Paketlänge sind jeweils **neu** zu bestimmen
 - Die **Headerprüfsumme** wird neu berechnet

- Ablauf der Defragmentierung:
 - Die Zielstation setzt die Fragmente eines Datagramms wieder zusammen
 - Die **Zusammengehörigkeit** entnimmt sie dem **Identifikationsfeld**
 - Die ankommenden Fragmente werden zunächst gepuffert
 - Bei Eintreffen des ersten Fragments wird ein **Timer** gestartet
 - Ist der **Timer** abgelaufen bevor alle Fragmente eingetroffen sind, wird alles **verworfen**
 - Im anderen Fall wird das Datagramm am N-SAP **zur Transportschicht hochgereicht**

Fragmentierung, Übung



Fragment 1

Rest des Headers		
Identifikation=120	Flags:MF = <u> </u>	FO = <u> </u>
Datenbyte 0 ... 375		

Fragment 2

Rest des Headers		
Identifikation= <u> </u>	Flags:MF = <u> </u>	FO = <u> </u>
Datenbyte <u> </u> ... <u> </u>		

Fragment 3

Rest des Headers		
Identifikation= <u> </u>	Flags:MF = <u> </u>	FO = <u> </u>
Datenbyte <u> </u> ... <u> </u>		

Fragment 1

Rest des Headers		
Identifikation=120	Flags:MF = 1	FO = 0
Datenbyte 0 ... 375		

Fragment 1

Rest des Headers		
Identifikation=120	Flags: MF = 1	FO = 0
Datenbyte 0 ... 375		

Fragment 2

376 / 8 = 47

Rest des Headers		
Identifikation=120	Flags: MF = 1	FO = 47
Datenbyte 376 ... 751		

Fragment 1

Rest des Headers		
Identifikation=120	Flags: MF = 1	FO = 0
Datenbyte 0 ... 375		

Fragment 2

376 / 8 = 47

Rest des Headers		
Identifikation=120	Flags: MF = 1	FO = 47
Datenbyte 376 ... 751		

Fragment 3

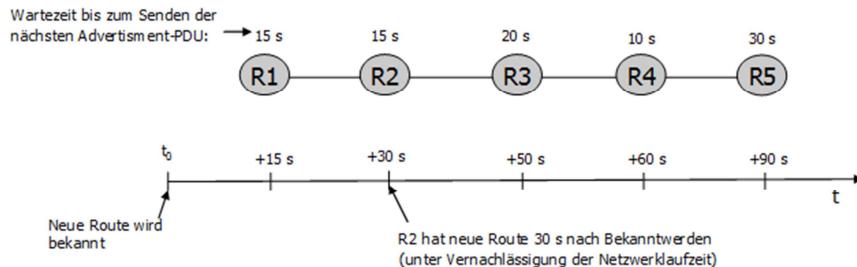
752 / 8 = 94

Rest des Headers		
Identifikation=120	Flags: MF = 0	FO = 94
Datenbyte 752 ... 999		

Routing in autonomen Systemen, RIP

- RIP (Routing Information Protocol) wurde ursprünglich von XEROX entwickelt
- Wird in kleinen AS immer noch stark verwendet
- Einfach und leicht zu implementierendes **Distance-Vector-Protocol**
- Als Metrik wird „**Hop-Count**“ verwendet
- RIPv1 ist **klassenorientiert** und ermittelt das Zielnetzwerk anhand der ersten Bits der Ziel-IP-Adresse
 - 0 = Klasse A, 10 = Klasse B, 110 = Klasse C, ...
- Implementierung unter Unix durch **routed**-Prozess
- Routing-Tabelle anschauen: **netstat -r**
- RIP versendet die Routing-Einträge alle 30 s in sog. Advertisement-PDUs
 - RIPv1 über MAC-Broadcast
 - RIPv2 über Multicast auf Subnetzebene
- Weitere Timer für das Deaktivieren (180 s) und Entfernen (240 s) von Routing-Einträgen definiert
- Nicht geeignet für WAN-Routing, eher im LAN wegen Broadcast/Multicast
- Max. 25 Routeneinträge pro RIP-Nachricht und es werden immer alle Routen übertragen
 - Ggf. mehrere RIP-PDUs senden
- **Konvergenzzeit:** Zeit, die erforderlich ist, bis alle Router die aktuelle Struktur eines Netzes kennen gelernt haben → Konsistentes Netzwerk
 - Ist bei RIP relativ lange, da RIP-Router Änderungen erst verarbeiten und dann an die Nachbarn propagieren

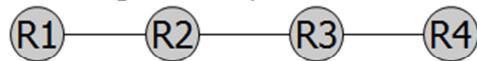
- **Split Horizon** ist eine Methode, um die Konvergenzzeit kürzer zu halten
- Anm: Max. 15 Hops wurden gewählt, um die Konvergenzzeit zu beschränken
- Konvergenzzeit und das Problem der langsamen Konvergenz
- Verbreitung von Routing-Tabellen-Einträgen in mehreren Takten a' 30 s bestimmt die Konvergenzzeit
- Bis zum nächsten Senden einer Advertisement-PDU dauert es im Mittel 15 s, zufallsabhängig



Beispiel:

- **Routing-Schleifen** („Count-to-Infinity-Problem“) möglich
 - Lösung 1: **Split-Horizon-Technik** → Routing-Tabellen enthalten **zusätzlich** die Info, woher die Routing-Info kommt
 - Lösung 2: **Poison-Reverse** (vergifteter Rückweg) → Alle Routen werden propagiert, aber zum Ursprungsnetz hin als „nicht erreichbar“ („vergiftet“)

a) Alle Verbindungen R1-R2, R2-R3 und R3-R4 intakt

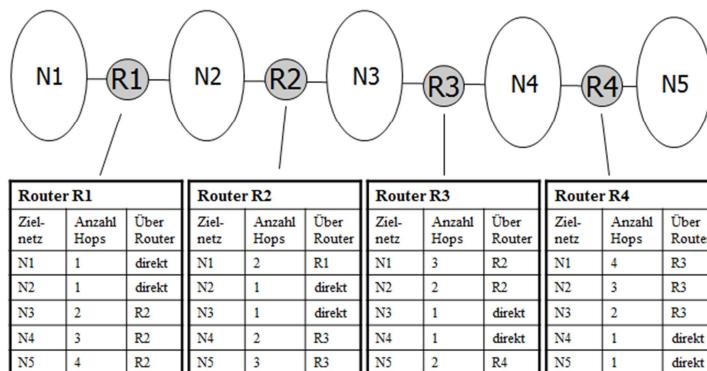


b) Verbindung R1-R2 fällt aus

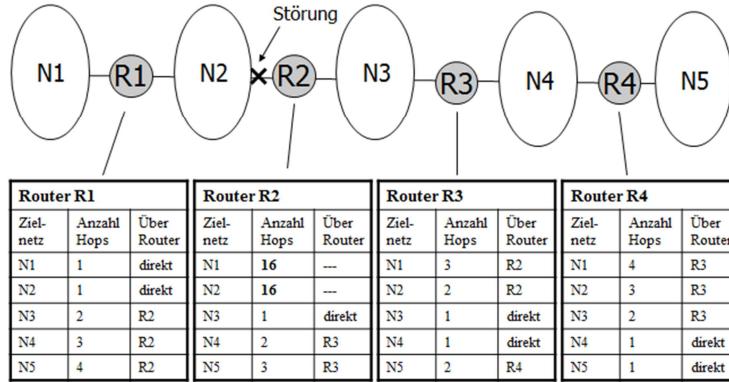


- Was passiert mit und ohne Split-Horizon?

- Routing-Tabellen im eingeschwungenen Zustand (nach einer angemessenen Konvergenzzeit)



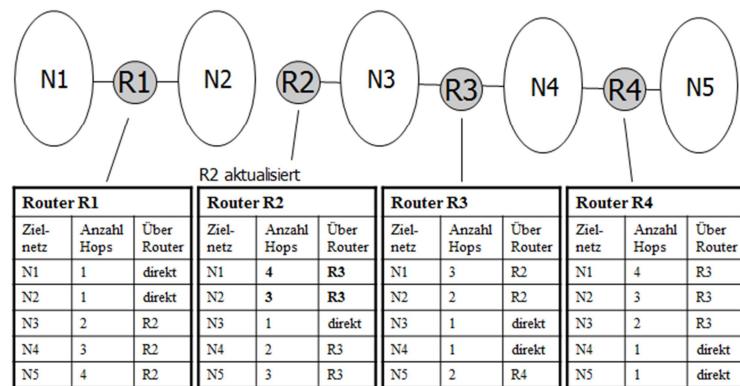
- Störung im Netzwerkzugang von R2 zu N2: R2 korrigiert sofort



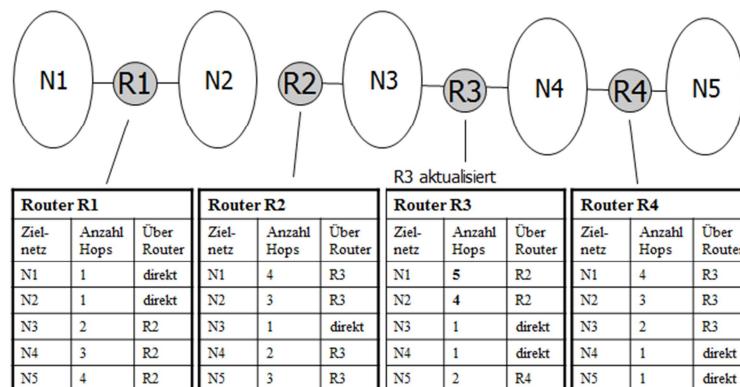
- **Ohne Split Horizon:**

- R3 hat noch die Routing-Info, dass N1 über drei Hops erreichbar ist
- R3 propagiert diese Info an R2, also an den Router, über den R1 erreicht wurde
- R2 glaubt dies und sendet Pakete zu R1 nun über R3
- Ping-Pong-Effekt, Routing-Schleife bis Hop-Count = 16, dann erst wird R1 als nicht erreichbar markiert
- Im Folgenden aus Sicht von R2 und R3 skizziert!

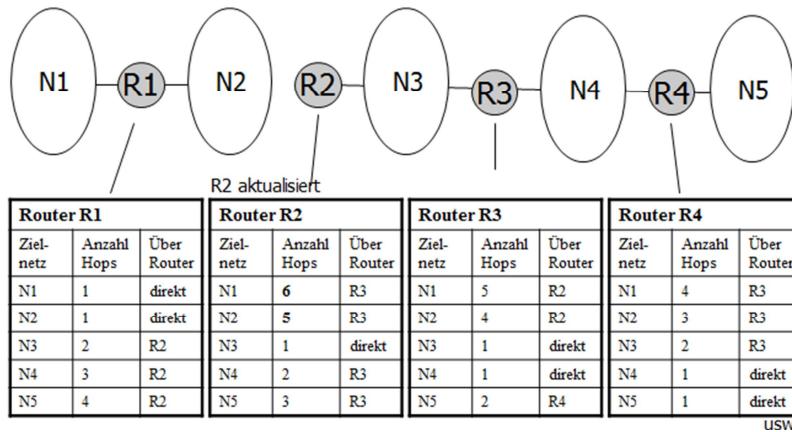
- Nun sendet R3 eine RIPv1-Advertisement-PDU an seinen Nachbarn R2 $\rightarrow \{(N1, 3), (N2, 2), (N3, 1), (N4, 1), (N5, 2)\}$



- Als nächstes sendet R2 eine RIPv1-Advertisement-PDU an seinen Nachbarn R3 $\rightarrow \{(N1, 4), (N2, 3), (N3, 1), (N4, 2), (N5, 3)\}$



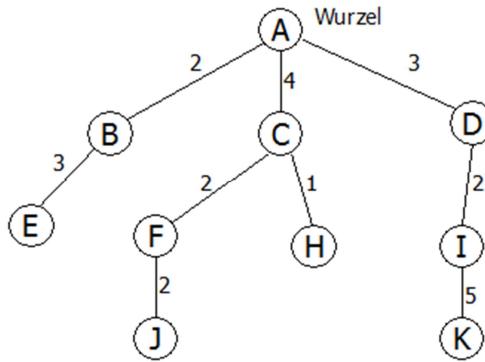
- Nun sendet R3 wieder eine RIPv1-Advertisement-PDU an seinen Nachbarn R2 $\rightarrow \{(N1, 5), (N2, 4), (N3, 1), (N4, 2), (N5, 3)\}$



- Mit Split Horizon:**
 - R3 weiß, woher die Routing-Info für N1 kommt (von R2)
 - Route mit höheren Kosten wird nicht zurückpropagiert
 - Keine Routing-Schleife (in diesem Fall)
- Metrik = 16 \rightarrow Netzwerkziel nicht erreichbar
- AFI = Adressierungsart, bei IP-Adressen immer 0x02
- Sonstiges zu RIP**
 - RIPv1 **unterstützt** CIDR/VLSM **nicht**, RIPv2 schon
 - Subnetzmaske wird in RIPv1 **nicht** übermittelt
 - Klasse wird aus den ersten Bits der Ziel-IP-Adresse ermittelt
 - RIPv2 **unterstützt** CIDR/VLSM
 - RIPv2 kann **Split-Horizon** und Split-Horizon mit Poisson-Reverse
 - RIPv2 kann selbst ausgelöste Router-Aktualisierungen (**Triggered Updates**) bei Ankunft einer Advertisement-PDU
 - Höhere, aber immer noch nicht perfekte Konvergenz, mehr Netzwerklast
 - RIPv1 kommuniziert über **Broadcast**, RIPv2 über **Multicast**

Routing in autonomen Systemen, OSPF

- OSPF ist **für große Unternehmensnetze** gedacht, für kleine wird noch RIP bzw. statische Routing-Tabellen verwendet
- Offener Standard** (Open SPF), RFC 1247 u. 2328
- OSPF ist ein **Link-State-Protocol**
 - „Link State“ ist der Zustand einer Verbindung zweier Router \rightarrow zustandsorientiert statt entfernungsorientiert (RIP)
- Kommunikation mit unmittelbaren, **designierten Nachbarn** zum Austausch der Routing-Information
- Jeder Router führt **eigene Datenbasis** (Link-State-Datenbank) mit allen Routing-Einträgen des Netzes
- Jeder IP-Router erzeugt aus seiner Sicht einen Spanning Tree (SPF-Baum) für das ganze Netzwerk
- Wurzel ist der Router selbst
- Verzweigung = günstigste Route



OSPF, Sonstige Features

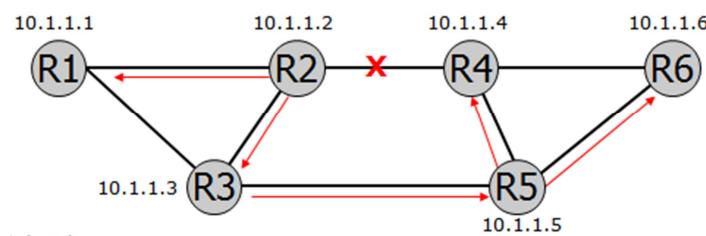
- **Load Balancing** bei Pfaden mit gleichen Kosten
 - gleichmäßige Verteilung, besser als bei RIP
- Nutzung spezieller **Multicast-Adressen** zur Kommunikation
- Unterstützung der Router-**Authentifizierung** zur Vermeidung von Angriffen (mehr Sicherheit)
- Metriken (Kosten) in RFCs nicht festgelegt
 - Cisco IOS (Internet Operating System) verwendet z.B. als Metrik die Gesamtbandbreite aller Ausgangsschnittstellen vom aktuellen Router bis zum Zielnetzwerk

Funktionalität

- Alle Router suchen beim Start ihre Nachbarn mit **Hello-PDUs**, aber nicht alle angrenzenden Router werden auch zu Nachbarn (sog. **adjacents**)
- **Zyklischer Abgleich** der Link-State-Datenbanken mit den Nachbarn
- **Lebendüberwachung** periodisch unter den Nachbarn
- **Link State Updates** (Konsistente Datenhaltung in allen Routern) periodisch **und** bei Topologieänderungen
- **Refreshing** spätestens alle 30 Minuten

OSPF Konvergenz

- Verteilung einer Veränderung im Netz geht schnell und Information wird vor der Verarbeitung weiterkommuniziert
 - Hohe Konvergenz
- Beispiel für Routen-Austausch:
 - Verbindung zwischen 10.1.1.2 und 10.1.1.4 fällt aus
 - Link-State Updates werden über das ganze Netz verteilt
 - Nachdem DB synchronisiert ist, gibt es nur noch eine kürzeste Route



OSPF, Routertypen

- Bei OSPF gibt es vier Router-Klassen
 - Interne Router der Area
 - Router an Bereichsgrenzen (Area-Grenzen)
 - Verbinden zwei oder mehrere Areas
 - Backbone-Router
 - Befinden sich im Backbone
 - AS-Grenz-Router (ASBR)
 - Vermitteln zwischen autonomen Systemen
- Bei Einsatz in Broadcast-orientierten LANs: Verwendung von sog. designierten Routern
 - Alle Router bauen eine Nachbarschaft (Adjacencies) zu diesem Router auf → Reduzierung der Kommunikation

OSPF, Nachbarschaften

- Nachbarschaften werden bei der Initialisierung aufgebaut
 - Senden von Hello-PDUs alle 10 s
 - Hello-PDUs enthalten bekannte Nachbar-Router
- Wenn Nachbarschaft aufgebaut ist, wird alle 40 s eine Hello-PDU als Heartbeat gesendet
 - Bleibt diese aus, wird Nachbar für ausgefallen erklärt
- Die Verteilung von Änderungen der Routing-Tabellen erfolgt immer zu allen Nachbarn

OSPF, PDUs

- Es gibt fünf OSPF-PDU-Typen:
 - **Hello:** Feststellung der Nachbarn, Aufbau von Nachbarschaften
 - **Database Description:** Bekanntgabe der neuesten Daten
 - **Link State Request:** Informationen vom Partner anfordern
 - **Link State Update:** Informationen an Nachbarn verteilen
 - **Link State Acknowledgement:** Bestätigung eines Updates
- PDUs werden direkt über IP gesendet, (siehe IP-Header, Protokoll = 89)
- Direkte Übertragung an Nachbarn oder über spezielle Multicast-Adressen
- OSPF-PDUs werden nur zwischen Nachbarn ausgetauscht
- Kein Weiterroute außerhalb des eigenen Netzes (IP-Header, TTL=1)

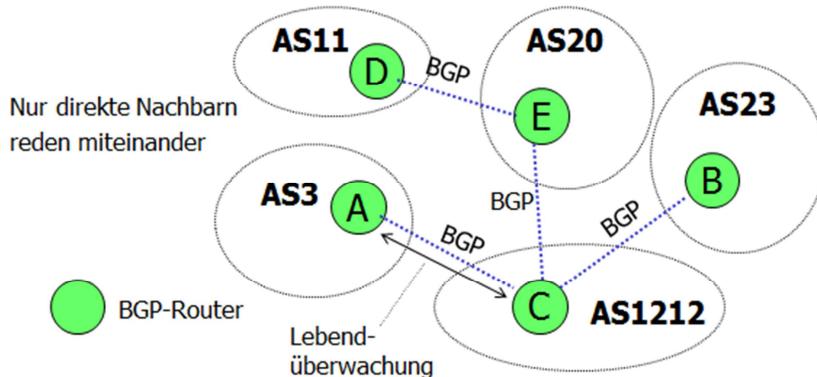
Übung

- Welche Routing-Informationen verwaltet RIP und welche OSPF?
 - RIP verwaltet als Distanz-Vektor-Protokoll die Entfernung (Anzahl Hops) und Richtung zum Ziel (Vektor)
 - OSPF verwaltet sog. Links in einer sog. Link-State-Datenbasis. In dieser wird die gesamte Topologie des Netzes verwaltet
 - Bei OSPF sind also alle Routen im Netz genau bekannt, bei RIP nicht
- Warum konvergiert OSPF im Vergleich zu RIPv2 besser?
 - Beide senden Veränderungen sofort (Triggered Update)
 - RIPv2 verarbeitet die Routing-Updates zuerst und dann werden sie an die Nachbarn weitergereicht, bei OSPF ist dies genau umgekehrt

- Wie werden bei RIP und bei OSPF die Kosten (Metrik) berechnet?
 - Bei RIP: Anzahl Hops ist die verwendete Metrik
 - Bei OSPF: Nicht festgelegt, Cisco nutzt z.B. die Bandbreite des Gesamtpfades als Metrik, je größer, umso besser. Dies geht, da die Gesamttopologie bekannt ist.

BGP

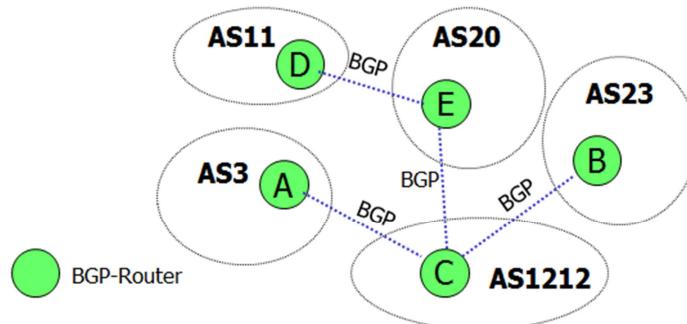
- BGP (= Border Gateway Protocol) ist ein EGP (RFC 1654), derzeit im Internet nur BGP im Einsatz
- BGP ermöglicht das Routing zwischen verschiedenen autonomen Systemen (AS)



- BGP ist ein **Pfadvektorprotokoll** (Path Vector Protocol), ähnelt dem Distance-Vector-Protokoll
- BGP-Router kennen die besten Route zu anderen AS als **vollständigen Pfad** (**auf AS-Ebene**)
- Jeder BGP-Router führt eine **Datenbank** mit Routen zu allen erreichbaren autonomen Systemen
- Routing-Tabellengröße:
 - ~ 200.000 Einträge

Routing über autonome Systeme hinweg, BGP

- Routing-Tabelle von D enthält folgende Routen:
 - AS11 – AS20
 - AS11 – AS20 – AS1212
 - AS11 – AS20 – AS1212 – AS23
 - AS11 – AS20 – AS1212 – AS3



- Ein BGP-Router informiert **periodisch** alle **Nachbar**-BGP-Router **genau** über die zu nutzenden Routen
 - UPDATE-PDUs werden versendet, sog. Advertisements
- Zyklen in Routen (Routing-Schleifen) werden bei Übernahme der Information geprüft
 - Eigene AS-Nummer darf nicht in Route sein
 - Count-to-Infinity-Problem tritt nicht auf
- BGP-Router überwachen sich gegenseitig (Heartbeat-Protokoll → KEEPALIVE-PDU)

- BGP-Router verwendet zur Auswahl der besten Routen eine **Routing-Policy**
- Routing-Policy (Regeln), Beispiele:
 - Kein Verkehr über einen bestimmten Knoten
 - Sicherheitsaspekte
 - Kostenaspekte
 - ...
- Eine Route, die die Regeln verletzt, wird auf „unendlich“ gesetzt
- BGP nutzt TCP (Port 179) als Transportprotokoll für seine Nachrichten (verbindungsorientiert!)

ICMP: Einführung

- ICMP (Internet Control Message Protocol, RFC 792)
 - Dient der Übertragung von unerwarteten Ereignissen und für Testzwecke
 - Beispiel 1: Ein Netzwerk ist nicht erreichbar: Ein IP-Router sendet in diesem Fall die ICMP-PDU „Network Unreachable“
 - Beispiel 2: Das ping-Kommando verwendet z.B. ICMP-PDUs (Echo Request, Echo Reply)
 - Beispiel 3: Das Kommando traceroute (tracert) nutzt ICMP (Typ=11, Time-to-live exceeded)
- ICMP-Nachrichten werden in IP-Datagrammen versendet
- ICMP-Beispiel: **Destination unreachable** → Ein Router kann ein Datagramm nicht ausliefern
- Router sendet ICMP-Nachricht an den Absender
- Code: 0 = Netzwerk nicht erreichbar, 1 = Rechner nicht erreichbar, ...

ARP

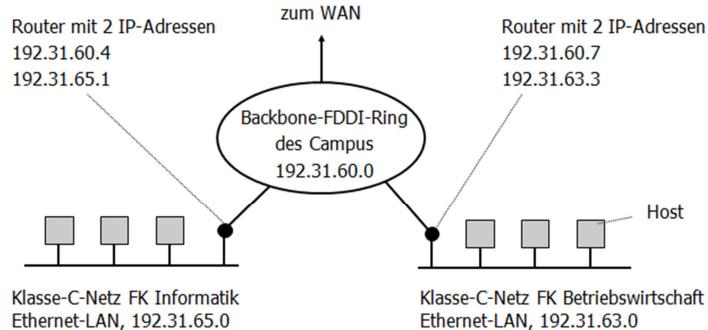
- ARP (Address Resolution Protocol), RFC 826
 - ARP dient dem **dynamischen Mapping** von IP-Adressen auf Schicht-2-Adressen (MAC-Adressen)
 - Jeder Host kennt seine eigene Schicht-2-Adresse, nicht aber die Adressen der anderen Hosts
 - Jeder Host führt einen **ARP-Cache** und merkt sich darin Schicht-2-Adressen, die über ARP im **IP-Broadcasting** erfragt werden können → **Periodisches Löschen vermeidet Inkonsistenz!**
 - Ist der Zielhost nicht gespeichert, wird ein **ARP-Broadcast** mit der IP-Zieladresse als Parameter versendet
 - Der Zielrechner antwortet mit einem **ARP-Reply** (MAC-Adresse)
 - IP-Router übernehmen Rolle des **ARP-Proxy**

RARP

- RARP = Reverse ARP
 - RARP wird verwendet, wenn die eigene IP-Adresse eines Hosts nicht bekannt ist, aber benötigt wird
 - RARP sendet RARP-Request mit der eigenen MAC-Adresse als Broadcast
 - Anwendungsfall:
 - Plattenlose Desktop-Arbeitsplätze, die beim Booten Ihre IP-Adresse ermitteln wollen

ARP: Beispiel

- Typisches Netzbeispiel: Campusnetz (nach Tanenbaum)



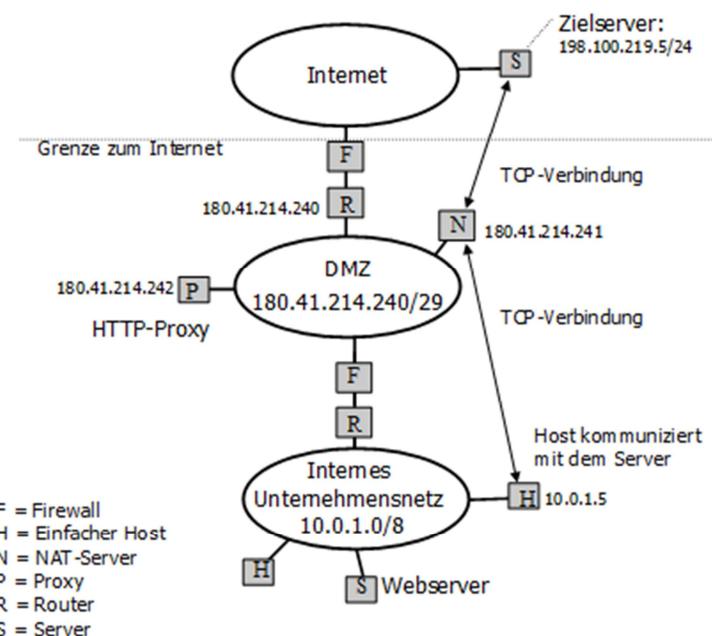
Network Address Translation (NAT): Einführung

- Bei NAT handelt sich vorwiegend um eine Möglichkeit, die **Sicherheit** im Unternehmensnetz zu erhöhen
- NAT dient auch dazu, Netzwerkadressen einzusparen
- Für ein Netz (Unternehmensnetz) benötigt man **nur noch eine bzw. wenige offizielle IP-Adresse**
- Intern kann dann eine beliebige, nach außen nicht sichtbare, Netzwerksnummer verwendet werden
→ Private IP-Adressen!

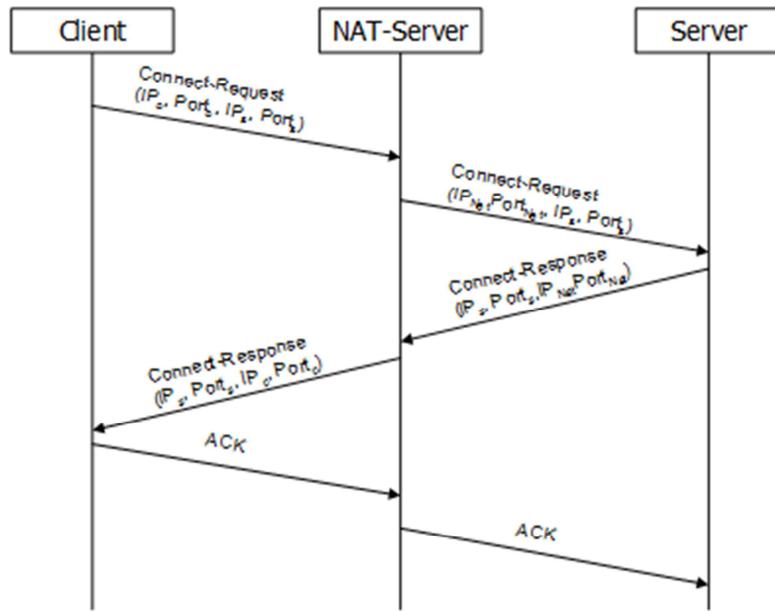
NAT, NAPT

- IP-Router bzw. NAT-Server „**mappen**“ bei NAT an kommende Pakete auf interne Hostadressen und umgekehrt
- IP-Router arbeiten nach außen als **Stellvertreter** (Proxies) für alle internen Hosts
- Verschiedene Varianten von NAT verfügbar, auch NAPT = Network Address Port Translation

NAT: Beispiel



NAT: Nachrichtenfluss

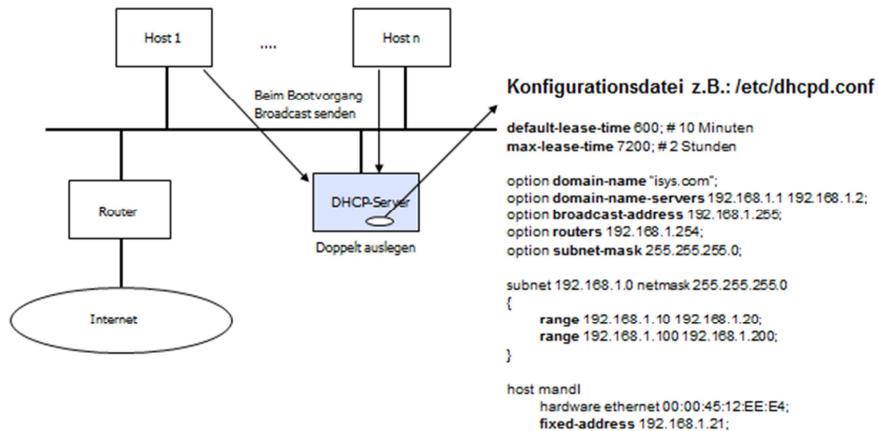


DHCP: Einführung

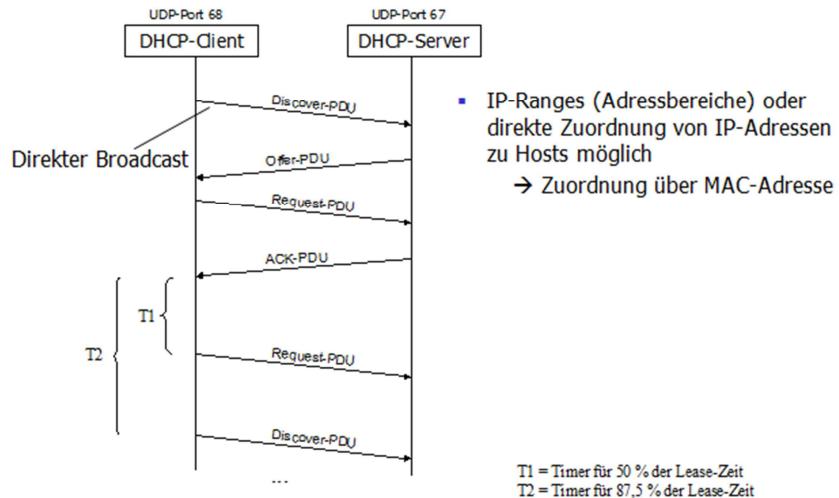
- Manuelle Netzwerkkonfiguration ist schon bei kleinen Netzen ein Problem
- Dynamic Host Configuration Protocol schafft Abhilfe
 - Hosts müssen Adressen nicht mehr kennen, sie werden dynamisch beim Booten besorgt
- Dynamische Vergabe von IP-Adressen und weiteren Netzwerk-Parametern über einen DHCP-Server:
 - Subnetzmaske
 - DNS-Server-Adresse
 - IP-Router-Adresse
 - ...

DHCP: Beispielnetz

- Meist beziehen nur Arbeitsplatzrechner Ihre IP-Adresse vom DHCP-Server



DHCP: Kommunikation beim Bootvorgang



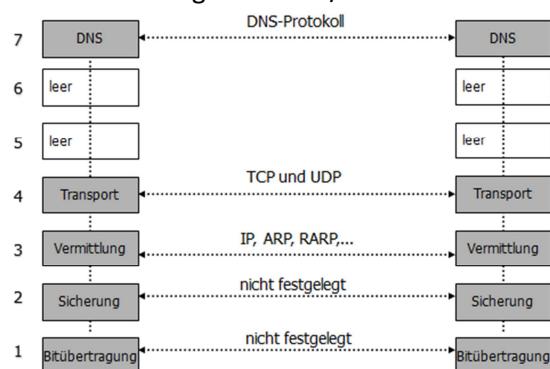
- Lease-Zeit = Nutzungszeit für IP-Adresse
 - Parameter (Timer) werden beim dyn. Konfigurieren an den Client gesendet
 - Parameter T1 gibt standardmäßig 50 % der Lease-Zeit an
 - Client sendet erneut einen DHCP-Request
 - Parameter T2 87,5 % der Lease-Zeit
 - Wenn kein ACK vom Server kommt, dann erneutes DHCP-Discovery durch Client

Wichtige Administrations-Kommandos

- Diagnose- und Konfigurationskommandos im TCP/IP-Umfeld, die man öfter mal braucht:
 - ping
 - hostname
 - netstat
 - nslookup (kommt später bei DNS)
 - arp
 - traceroute (Windows: tracert)
 - ifconfig
 - route
 - ipconfig (Windows)
 - nbtstat (Windows)

Fallstudie DNS

Einordnung in die TCP/IP-Protokollfamilie



Domain Name System (DNS):Hintergrund

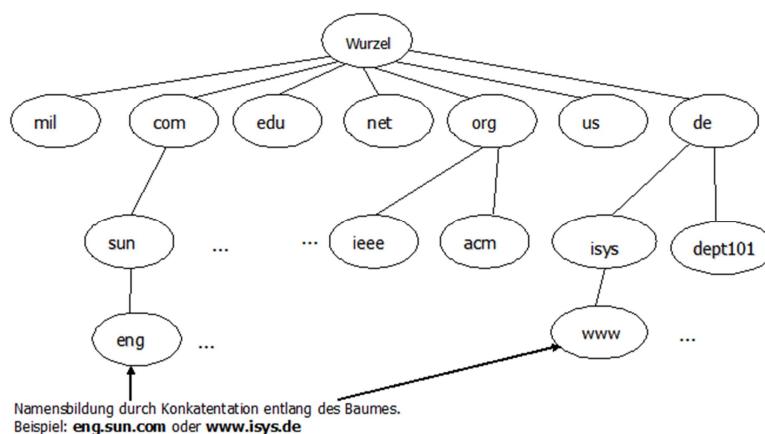
- Im ARPANET mit einigen hundert Hosts war die Verwaltung der Adressen noch einfach
 - Es gab eine Datei **hosts.txt** auf einem Verwaltungsrechner, die alle IP-Adressen enthielt (flacher Namensraum)
 - Die Datei wurde nachts auf die anderen Hosts kopiert
- Als das Netz immer größer wurde, führte man DNS ein
- DNS dient prinzipiell der **Abbildung von Hostnamen auf eMail- und IP-Adressen**
- DNS ist ein Internet **Directory Service**

Domänen und Subdomänen

- DNS ist in den RFCs 1034 und 1035 definiert
- DNS ist ein hierarchischer Namensverzeichnis für IP-Adressen (Adressbuch des Internets)
- DNS verwaltet eine **Datenbank**, die sich über zahlreiche Internet-Hosts erstreckt
- Konzeptionell ist das Internet in **mehrere hundert Domänen** aufgeteilt
- Die Domänen sind wiederum in **Teildomänen (Subdomains)** untergliedert, usw.

DNS-Namensraum

- Weltweit verteilter Namensraum
- **Baum**, an dessen Blättern dann die Hosts hängen (**rein organisatorisch**, nicht physikalisch)



- DNS ist eine baumförmige **weltweite Vernetzung** von Name-Servern
- DNS bildet eine weltweit verteilte Datenbank (**DNS-Datenbank**)
- Jeder Knoten im DNS-Baum stellt eine Domäne dar und kann mit einem Verzeichnis in einem Dateisystem verglichen werden
- Jeder Knoten hat einen Namen

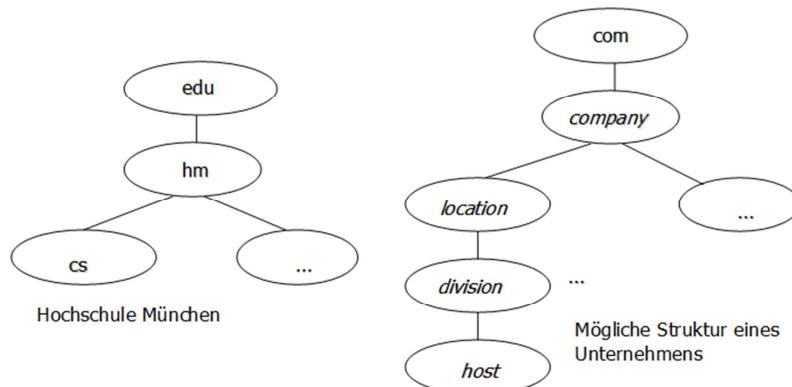
Root-Name-Server

- Es gibt derzeit **weltweit 13 sog. Root-Name-Server (A...M)**
 - 10 in Nordamerika, 1 in Stockholm, 1 in London, 1 in Tokio
- Root-Name-Server geben Informationen über die weitere Suche
- Ein Root-Name-Server
 - verfügt über eine Referenz-Datenbank aller von der ICANN freigegebenen Top-Level-Domains (TLD) und
 - die wichtigsten Referenzen auf die sog. Top-Level-Domain-Server
- Ein Root-Name-Server kennt somit immer einen DNS-Server, der eine Anfrage beantworten kann

- DNS erlaubt DNS-PDUs (UDP) bis zu einer Größe von 512 Byte
 - Daher max. 13 DNS-Records in einer PDU
 - EDNS (Extended DNS) als DNS-Erweiterung ermöglicht die zehnfache Größe (RFC 2671)
- A und J haben **zentrale** Bedeutung. Sie verteilen die Datenbasis für alle anderen Root-Name-Server zweimal täglich
- Root-Name-Server bestehen aus z.T. **vielen Einzelservern**:
 - Root-Name-Server F besteht aus 33 Serverrechnern (2009)
 - Root-Name-Server K besteht aus 16 Serverrechnern (2009)
- Diese Root-Name-Server sind auf der ganzen Welt verteilt und einige nutzen heute einen **Anycast-Mechanismus** (eine IP-Adresse) zur Datenverteilung

DNS-Namensraum

- **Beispiele** für Domänen mit untergeordneten Subdomänen



Name Server und Zonen

- DNS-Name-Server verwaltet jeweils **Zonen** des DNS-Baums, wobei eine Zone an einem Baumknoten beginnt und die darunter liegenden Zweige beinhaltet
- Ein Name-Server (bzw. die entspr. Organisation) kann die Verantwortung für Subzonen an einen weiteren Name Server **delegieren**
- Die Name-Server kennen jeweils ihre Nachbarn in der darunter- oder darüberliegenden Zone
- Informationen des DNS werden in sog. **Resource Records** verwaltet

Autoritative Name Server und andere...

- **Autoritativer Name Server:**
 - Verantwortlich für eine Zone
 - Mind. einer muss in Zone sein (Primary Nameserver)
 - Kennt alle Adressen der Zone genau
 - Siehe Konfigurationsdatei (**SOA-Resource-Record**)
- **Nicht-autoritativer Name Server:**
 - Bezieht Informationen von anderen Servern
 - Verfalldatum über TTL-Parameter geregelt

Zonentransfer zwischen Primary

Name Server

- In jeder Zone muss es mindestens zwei Name Server geben (**primary und secondary**), die auch miteinander kommunizieren
- Für normale Anfragen wird ein **UDP-basiertes** Protokoll verwendet, größere Abfragen über TCP
- Primary und Secondary kommunizieren über **TCP**
- Name Server werden von Internet Service Providern (ISP) und Netzbetreibern betrieben
- Die Domain **.de** verwaltet das Deutsche Network Information Center (**DENIC**)
 - betrieben in Frankfurt (früher TU Dortmund, dann TU Karlsruhe)

Domainnamen

- DNS-Namen bestehen aus zwei Teilen, einem Hostnamen und einem Domänennamen, die zusammen den „**Fully Qualified Domain Name**“ (FQDN) bilden
 - Max. 255 Zeichen, keine Sonderzeichen, Umlaute oder Leerzeichen
 - Max. 63 Zeichen pro Teildomäne oder Hostname
 - Beispieladresse: www.dept101.com (dept101.com ist der Domänenname)
- Zu jeder Domäne gehört eine Körperschaft, die diese verwaltet und für die Namensvergabe zuständig ist, oberste Körperschaft ist das NIC

Zusammenfassung: Das Wichtigste in Kürze

- Domains, TLDs (Top-Level Domains)
- Zonen
- Resolver
- DNS-Server, Root-Name-Server, TLD-Server
- Autoritative versus nicht-autoritative Server
- Rekursive und iterative Namensauflösung