Rechnernetze I

Adrian Immanuel Kieß

27. März 2018 / Version 0.9 / Revision 2018.03.27//19:01

http://www.kiess.onl

- Dienstags, 13:15Uhr, Jahnallee – Hörsaal Süd
- Buch zur Vorlesung: Computer Networking, Jim Kurose/Ross, 3rd ed. Pearson 2002. Buch verfolgt Top-Down-Ansatz.
- Folien von Kurose/Ross via Website verfügbar (nur via UNI-Account)
- 2 SWS, 4 Übungsblätter, 2 *Großübungen* von Lars Lattig
- Mobile Ad-Hoc-Netze wird *aufbauend* auf dieser Vorlesung angeschlossen
- Prüfung evtl. mündlich, falls Teilnehmerzahl ≤ 30

Stichpunkte zur Vorlesung

1. Vorlesung

- *P2P:* Verschiebt Rechenleistung an die Kanten
- Zugangsnetze
- Cellular Networks
- Klassische Netzapplikation: Client-Server-Applikation
- Von Relevanz für Informatiker: (Schicht 3), Schicht 4 (Datensatzbau), Schicht 5 (Anwendung) ISO/OSI
- SYN $\rightarrow$ ACK $\rightarrow$ SYN/ACK (Verbindungsaufbau, TCP-Handshake)
- Netzkanten: Anwendungen
- Endknoten: Applikationen
- Schichten 1-3: Router
- Client/Server: Single-Point of Failure, anders als P2P
- P2P: Keine Abhängigkeit von einem zentralem Server
- Verteilte Hashtabellen: aktuelles Forschungsgebiet
- FDMA: Aufteilung nach Frequenz
- TDMA: Aufteilung nach Zeit

Vorlesung Rechnernetze I – WS 2005/06

Adrian Immanuel Kieß < adrian@kiess.onl > http://www.kiess.onl

2. Vorlesung

- Tier-1 ISPs / peering (Telekommunikationsgesellschaften)
- Tier-2 ISP (customer Tier-1 ISP)
- Tier-3 ISP (local ISP)
- (vor zehn Jahren. . .: Übertragunsrate niedriger. :))
- Pufferüberlauf (NIC, Router, etc pp.)
- Leitungsvermittelt / Paketvermittelt
- 'Nobeldisko'; $2/3$ Optimalauslastung – Zugangskontrolle (hinreichend unwichtig)
- Queueing delay (Bäckereibedienung $\rightarrow$ einpacken von Brötchen) [free/dropping]
- Delay (Packet-switching): transmission, propagation, modal processing, queueing (traceroute)
- Q-Delay: La/R ~ 0: average; La/R $\rightarrow$ 1: large delay; La/R > 1: more than can be served
- Anwendungen sind das Salz in der Suppe
- Katastrophenschutz (Menge der Aktionen ist nicht endlich)
- Layering / Schichten (Reifen unrelevant für Motor)
- Internet protocol stack: application $\rightarrow$ transport $\rightarrow$ network $\rightarrow$ link $\rightarrow$ physical (TCP *zuverlässig*; UDP unzuverlässig)

3. Vorlesung

- Protokolle (HTTP, FTP. . .)
- Prozess: Programm das auf Host läuft.
- Unterschied zw. Applikation u. Apllikationslayer
- *Internet* $\rightarrow$ Hosts, *WWW* $\rightarrow$ Webseiten
- Typische Netzapplikation besteht aus Server und Client.
- Protokolle spezifizieren, wie Nachrichten formatiert sein müssen.
- Jeder Host besitzt eindeutige Adresse.
- Transportservice? Datenverlusst, Verzögerung (Sensitivität), Bandbreite. (z.B. Bahn / Flugzeug)
- Protokolle: TCP und UDP. TCP: Verbindungsorientiert, reliable transport, Flusskontrolle, Congestionkontrolle. UDP wird hptsl. wegen opt. Bandbreite und Timing benötigt.
- Webseite besteht aus Objekten welche per URL adressierbar sind.
- HTTP ist *stateless*. Unterschied zw. HTTP 1.0 (nonpersistent, GET, POST, HEAD) und 1.1 (persistent, +PUT, +DELETE).
- RTT (round trip time). One RTT to initiate TCP connection; One RTT for HTTP request. Ergibt 2 x RTT + transmit time.

Vorlesung Rechnernetze I – WS 2005/06

Adrian Immanuel Kieß < adrian@kiess.onl > http://www.kiess.onl

- HTTP response message, status codes (200→OK, 301→moved, 404→not found, etc. pp.)
- COOKIES: keeping state. Caching: client-side. (object modified, 304→not modified)
- EMAIL: SMTP; user agent; mail servers. messages must be in 7-bit ASCII.

5. Vorlesung (Zuverlässiger Datentransfehr, Zustände)

- Rechner →Endgeräte oder Router
- UDP unzuverlässig; versenden von kurzen Nachrichten
- UDP → Post, TCP → DHL (Lieferung innerhalb einer Zeitschranke)
- IPv4; UDP → möglichst wenig Verwaltungsaufwand; Reihenfolge nicht garantiert → Anwendungsprozess muss Datenpakete sortieren. Kein Handshaking. Spart Verwaltungsaufwand, kein Zustand zu verwalten.
- UDP CHECKSUM → Fehlererkennung
- Protokolle sind Zustandsautomaten
- Recovery von Fehlern (ACK, NAK)
- ACK/NAK corrupted? sender add sequence number or retransmits; receiver discards
- KRIEG → Leute vermisst, und *tauchen dann doch plötzlich wieder auf.*
- FEHLERBEHANDLUNG: Timeouts (angemessene Zeit [nach fünf Jahren ohne Meldung kann man Personen als tot erklären lassen], TCP → Regelungsverkehr)

6. Vorlesung

- GBN: receiver extended FSM
- duplicate acknowledgement
- selective repeat: sendet Paket nur neu, dass nicht erhalten wurde
- connection-oriented transport: segement structure, realiable data transfer, flow control, connection management
- TCP → Timeout zu klein: zu viele Timeouts; zu gross: zu langsame Reaktion
- RTT variieren, aber sie müssen *Smooth Operator* spielen.
- Letzten zehn gemessenen RTT x 0.9 $\alpha = 0.125$
- 0.2 Sekunden pro Paket = nicht sichtbar, ab 1 Sekunde merkbare Verzögerung
- cumulative acks, pipelined segmens (3 dupl. ack = assume data loss after ack)
- Autoverkauf: Zwischenspeicher ist Autolagerhalle/Verkausfläche. Verlauf zum Konsum muss geregelt werden →FLUSSKONTROLLE.

Vorlesung Rechnernetze I – WS 2005/06

Adrian Immanuel Kieß < adrian@kiess.onl > http://www.kiess.onl

7. Vorlesung (Ende Datentransferprotokolle)

- Karawane von 100 LKW (Verkehr auf der Straße→Bandbreite)
- TCP → Etabliert Verbindung vor Datenaustausch
- Sequenznummern werden zum aufspüren von doppelten Nachrichten gebraucht (Siehe SYN/ACK)
- 3-Wege-Handshake
- flow control → Flusssteuerung
- Bedienzeit → Einkaufen in Selbstbedienungsladen; Samstag abends (e.g. 5 Personen im Laden): Wartezeit, Montag morgens (e.g. einzige Person im Laden): kaum Wartezeit.
- causes of congestion → one router, one buffer. sender retranmission if lost packet
- Ende-Zu-Ende Überlastkontrolle → Expilzites Feedback vom Netz
- Überlastkontrolle →Analogie zum Straßenverkehr
- ATM ABR (av. bit rate), RM (resource management) → hat sich wegen zu hohen Kosten nicht durchgesetzt
- TCP AIMD
- TCP Slow Start (When connection starts, increase rate exponentially)
- 3 dup ACKs indicates total overload
- TCP Tahoe, TCP Reno
- TCP Fairness
- **Delay modeling**

Übungen

1. Übung

1. 1.2
   (a) a) Bei der Leitungsvermittlung werden die Daten über eine fest zugewiesene Leitung übermittelt, während die Paketvermittlung die Daten in separaten Stücken überträgt. Die Leitungsvermittlung reserviert Ressourcen, die einer Verbindung zugewiesen werden.
   (b) Vor- und Nachteile Leitungsvermittlung
      i. + Garantierte Bandbreite für eine Verbindung
      ii. - Verbindungsaufbau notwendig
      iii. - Ungenutze Ressourcen bei Leerlauf
   (c) Vor- und Nachteile Paketvermittlung
      i. + Ressourcenteilung erlaubt die Ausnutzung der vollen Bandbreite und den Zugriff durch eine höhere Zahl von Nutzern
      ii. + Kein Verbindungsaufbau notwendig
      iii. - Keine garantierte Bandbreite

Vorlesung Rechnernetze I – WS 2005/06

Adrian Immanuel Kieß < adrian@kiess.onl > http://www.kiess.onl

iv. - Überlauf kann zu Verzögerungen und Verlust von Paketen führen, so dass spezielle Protokolle notwendig sind

2. 1.3
   (a) a) Ein leitungsvermitteltes Netzwerk wäre besser geeignet, da aufgrund der Aktivität über einen längeren Zeitraum der Aufwand für den Verbindungsaufbau vernachlässigbar ist. Außerdem kommt es zu einer verhersagbaren, gleichmäßigen Bandbreiteitenausnutzung. Für eine bekannte Übertragunsrate kann dann eine entsprechende Reservierung vogenommen werden, ohne viel Bandbreite zu verschwenden.
   (b) b) Eine Überlastkontrolle ist nicht notwendig. Selbst wenn alle Anwendungen auf einmal über die gleiche Leitung übertragen wollen, würde die Kapazität per Definition ausreichen. Kleine Puffer werden eventuell benötigt, weil nicht alle Parteien gleichzeitig auf das Medium zugeifen können.
3. 1.4
   (a) a) Paketvermittlung mit VC:
      i. - Zeit für die Übertragung eines Pakets über eine Leitung: $\frac{L+h}{R}$
      ii. - Auslieferung des ersten Pakets am Ziel (nach Q Verbindungen): $Q * \frac{L+h}{R}$
      iii. - Alle (L+h)/R Sekunden trifft ein weiteres der $M-1$ verbliebenen Pakete ein und die Setupzeit muss berücksichtigt werden: $t_s + (Q + M - 1) * \frac{L+h}{R}$
   (b) b) Paketvermittlung mit Datagrammen und verbindungslosem Dienst:
      i. - Setupzeit entfällt, aber Headergröße verdoppelt sich: $(Q + M - 1) * \frac{L+2h}{R}$
   (c) c) Paketvermittlung ohne Message Segmenting:
      i. - Zeit für die Übertragung der Nachricht über eine Verbindung: $\frac{L*M+2h}{R}$
      ii. - Zeit für die Übertragung über Q Verbindungen: $Q * \frac{L*M+2h}{R}$
   (d) d) Leitungsvermittlung:
      i. - Zeit für die Übertragung der Datei (keine store-and-forward delays, aber Enrichtungszeit): $t_S + \frac{L*M+h}{R}$
4. 1.5
   (a) a) Bei Leitungsvermittlung werden 10 Nutzer unterstützt, da für jeden Nutzer die von ihm benötigte Bandbreite (100kbps) reserviert wird.
   (b) b) Wahrscheinlichkeit für n aktive Nutzer:
5. 1.6
   (a) Die fünf Schichten lauten (top-down):
   (b) (. . .)
6. 2.1
   (a) WWW → HTTP
   (b) Filetransfer → FTP
   (c) E-Mail → SMTP
   (d) Login → Telnet
   (e) News → NNTP
7. 2.2

(a) a) IP-Adresse des Ziel-Hosts und die Portnummer des entspr. Sockets.

(b) b) Bei pers. HTTP ohne Pipelining wartet der Browser zunächst auf eine HTTP Antwort des kontraktierten Servers, bevor er eine HTTP Anfrage stellt. Wenn Pipelining genutzt wird, stellt der Browser bei Bedarf direkt eine neue Anfrage ohne eine Antwort abzuwarten. Pipelining →HTTP/1.1.

(c) c) FTP verwendet zwei parallele TCP Verbindungen (Steuerinf. u. Daten) →*"out-of-band"*

8. 2.3

(a) Die Nachricht wird von Alice's Host zu ihrem Mailserver per HTTP gesendet. Der Mailserver überträgt die Nachricht an Bob's Mailserver über SMTP. Bob transferiert die Nachricht dann von seinem Mailserver auf seinem Host über POP3.

9. 2.4

(a) a) DNS bietet neben der Umwandlung von Hostnamen in IP-Adressen noch folgende Funktionen:

i. - Host-Aliasing
ii. - Mail-Server-Aliasing
iii. - Lastverteilung/Caching

(b) b) Es ist möglich, dass der Mailserver und der Webserver einer Organisation den gleichen Aliasnamen für einen Hostnamen haben? Der RR-Typ (resource record) MX wird verwendet, um den Hostnamen des Mailservers auf seine IP-Adresse abzubilden.

10. 2.5

(a) a) Server

i. - Erzeuge einen Socket für ankommende Anfragen (welcomeSocket)
ii. - Warte auf ankommende Verbindungsanfragen
iii. - Bei Eingang einer Verbindungsanfrage wird ein neuer Socket zur Kommunikation mit dem anfragenden Client erzeugt (connectionSocket)
iv. - Austausch von Daten (Lesen von Anfragen, Schreiben von Antworten) über den Verbindungssocket
v. - Schließen des Verbindungssocket

(b) b) Client

i. - Erzeuge lokalen TCP-Socket (clientSocket) und Stelle TCP-Verbindung zum Server her (Anfrage von IP-Adresse und Portnummer des Prozesses auf dem Server)
ii. - Austausch von Daten (Senden von Anfragen, Lesen von Antworten) über den lokalen Socket
iii. - Schließen des lokalen Socket

(c) c) Mit Hilfe von Threads kann ein Server mehrere Client-Anfragen gleichzeitig bearbeiten

(d) d) Der Server benötigt $N + 1$ Sockets bei N gleichzeitigen Verbindungen, da er einen Socket pro Verbindung zum Austausch von Daten benötigt sowie einen Welcome-Socket zum Entgegennehmen von Verbindungswünschen. Ein UDP-Server kommt mit einem Socket aus. Es wird kein Welcome-Socket benötigt und alle Daten von verschiedenen Clients werden über diesen einen Socket empfangen.