Paul Masch und Benjamin Königsberg

**Netzwerke I:** Praktikum 1

29.10.2016

**Aufgabe 1.1 Netzwerk-Traces mit Wireshark**

Zu a) DNS -> Application Layer

* UDP -> Transport Layer
* IP -> Network Layer
* Ethernet II -> Data Link/Physical Layer

TCP -> Transport Layer

* IP -> Network Layer
* Ethernet II -> Data Link/Physical Layer

HTTP -> Application Layer

* TCP -> Transport Layer
* IP -> Network Layer
* Ethernet II -> Data Link/Physical Layer

Zu b) DNS -> Application Layer

* UDP -> Transport Layer
* IP -> Network Layer
* Ethernet II -> Data Link/Physical Layer

ICMP -> Network Layer

* IP -> Network Layer
* Ethernet II -> Data Link/Physical Layer

Zu c) ARP -> Physical Layer

DNS -> Application Layer

* UDP -> Transport Layer
* IP -> Network Layer
* Ethernet II -> Data Link/Physical Layer

TCP -> Transport Layer

* IP -> Network Layer
* Ethernet II -> Data Link/Physical Layer

FTP -> Application Layer

* TCP -> Transport Layer
* IP -> Network Layer
* Ethernet II -> Data Link/Physical Layer

FTP Server Software –> vsFTPd 2.0.7

**Aufgabe 2.1: HTTP Protokoll**

1A) **Absolute URL:** a.fsdn.com/sd/topics/linux\_64.png

1B) **Accept-Encoding: gzip, deflate** – Ja, er kann gzip gepackte Daten entpacken.

1C) **Connection: keep-alive** - Er möchte eine persistente Verbindung.

1D) **If-none-matching:** Ist ein Hash Wert einer angefragten Datei, den der Server mit dem Name E-Tag an den Clienten schickt. Bei erneuter Anfrage wird der Hash wert mit den Namen If-none-matching an den Server gesendet sind die Hash-Werte gleich wird die Datei nicht erneut gesendet.

1E)

HTTP/1.1 200 OK

Date: 22.10.2015 15:36:59 GMT

Expires: -1

Cache-Control: private, max-age=0

Connection: keep-alive

Content-Type: image/png; charset=ISO-8859-1

Content-Encoding: gzip

Server: gvs 1.0

Content-Length: 800

x-content-type-options: nosniff

HTTP/1.1 304 Not Modified

Date: 22.10.2015 15:36:59 GMT

Cache-Control: private, max-age=0

Connection: close

Content-Type: image/png; charset=ISO-8859-1

Content-Encoding: gzip

Server: nginx

Content-Length: 800

x-content-type-options: nosniff

**Blatt 3:**

**Aufgabe 3.1: Messung der Round Trip Time mit ping**

**a)** (siehe Anhang)

**b)**

In Netzwerken wird zwischen Ausbreitungsverzögerung, Übertragungsverzögerung, Verarbeitungsverzögerung und Warteschlangenverzögerung unterschieden.

Mit dem Programm ping wird die Round-Trip Time gemessen.

**c)**

Es gibt keine Abhängigkeit.

**Aufgabe 3.2:** **Verzögerungen bei Kommunikation über Zwischenstationen**

**a)**

**Verbindung von A nach R1:**

**DSL-uplink-Signalausbreitungsgeschwindigkeiten =**

**R** = 384.000bit/s

1Byte = 8bit

Man sendet also pro Sekunde auf eine Länge von

**Verbindung von R1 nach R2:**

**Satellitenverbindung-Signalausbreitungsgeschwindigkeiten =**

**R** = 2Mbit/s = 2.000.000bit/s

Man sendet also auf eine Länge von pro Sekunde

**Verbindung von R2 nach B:**

**100BaseT-Signalausbreitungsgeschwindigkeiten =**

**R** = 100Mbit/s = 100.000.000bit/s

Man sendet also auf eine Länge von pro Sekunde

**b)**

**Paketlänge L =** 64byte\*8 = 512bit

**Übertragungsrate =** 384.000

**Distance =** 3.3km = 3300m

**Distance =** 98.000km = 98.000.000m

**Distance =** 25m

**Signal-Propagation =**

**Signal-Propagation =**

**Transmission-Delay =**

**Propagation-Delay =**

**Processing-Delay =** 0

**Queueing-Delay =** 0

**Für 64byte:**

**Total-Delay =**

0,328s

**Für 10MB =** 83.886.080bit:

**Total-Delay =**

**c)**

Paketgröße 1 KByte

A zu R1

R1 zu R2:

R2 zu B:

Ende zu Ende für 1Kbyte = 0,35159s

Paketgröße 10MByte – 1KByte = 9999 KByte

Paket von A zu R1:

= 218,432s

Gesamtzeit = 1KByte (Ende zu Ende) + 9999KByte(im ersten Abschnitt, da der langsamste, welcher die Gesamtgeschwindigkeit bestimmt)

218,432+0,35159 = 218,784s

**d)**

Paketgröße 1 KByte

A zu R1

R1 zu R2:

R2 zu B:

Ende zu Ende für 1Kbyte = 0,02593812s

Paketgröße 10MByte – 1KByte = 9999 kByte

Paket von A zu R1:

= 218,432s

Gesamtzeit = 1KByte (Ende zu Ende) + 9999KByte(im ersten Abschnitt, da der langsamste, welcher die Gesamtgeschwindigkeit bestimmt)

=0,02593812s + 218,432s= **218,458s**

**e)**

A zu R1

R1 zu R2:

R2 zu B:

Ende zu Ende für 40Kbyte = **0.0059s**

**Paketrest 1 Kbyte – 40Byte = 960Byte**

**Pakete von A zu R1:**

Ende zu Ende für 960 Byte = **0.0200165s**

**Aufgabe 3.3: HTTP Performance**

RTT = 2\*(Propagation Delay + Processing Delay + Queueing Delay)

**a) Non-Persistent Connection:**

**1. Aufbau der TCP Verbindung** = 1RTT = 250ms

**2. Sende GET-Anfrage für HTML :** 1RTT + Transmission Delay

-> **Reply Delay** = RTT + Transmission Delay = 250ms + 1ms= 251ms

**3.Schließen der TCP Verbindung**

**4. Verarbeiten der HTML =** 0ms

**5. Schritt 1-4 10 mal für jedes Bild**

* **Total Delay =** 11\*(250ms+251ms) = 5511ms

**b) Parallel Connections:**

**1. 1 Verbindungen wird geöffnet =** 250ms

**2. GET-Anfrage für HTML=** 1RTT + Transmission Delay = 251ms (siehe a)

**4. Verarbeiten der HTML =** 0ms

**5. Öffne 10 Verbindungen für die Bilder =**250ms

**6. GET-Anfrage von allen Verbindungen für Bilder = 1RTT + Transmission Delay**

Transmission Delay = =10ms

Replay-Delay = 250ms+10ms = 260ms

**Total Delay =** 250ms+251ms+250ms+260ms=1011ms

**c) Persistent Connection:**

**1. 1 Verbindung wird geöffnet =** 250ms

**2. GET-Anfrage für HTML =** 1 RTT + Transmission Delay

-> **Reply-Delay** = RTT+ Transmission Delay = 250ms + 1ms= 251ms

**3. Verarbeiten der HTML:** 0ms

**4. Schritt 2, 10 mal für jedes Bild**

**5. Schließen der TCP Verbindung**

**-> Total Delay =** 250ms + 11\*(251ms) = 3011ms

**d) Persistent Connection mit Pipelining**

**1. 1 Verbindung wird geöffnet =** 250ms

**2. GET-Anfrage für HTML** 251ms

**3. Verarbeiten der HTML:** 0ms

**4. Alle GET-Anfragen für Bilder: 1 RTT + Transmission Time aller Bilder**

**Reply-Delay =** RTT + Transmission Delay =250ms + 10ms = 260

**-> Total Delay =** 250ms + 251ms +260ms = 761ms

**Aufgabe 3.4: Schichtenmodell**

**a) Vergleich ISO/OSI- und das Internet-Schichtenmodell**

**Das Internet-Schichtenmodell** fasst die sieben Schichten des **ISO/OSI-Schichtenmodells** in vier Schichten zusammen. Damit ist das ISO/OSI Modell deutlich flexibler, da es die Zusammenfassung und Entfernung von einzelnen Schichten zulässt. Beim Internet-Schichtenmodell sind die Protokolle fest an die Schichten gebunden und lassen deshalb keine Anpassung zu. Die Netzwerk-Protokolle TCP/IP sind fest im Internet-Schichtenmodell verankert und lassen sich nicht ersetzen. Nur die Anwendungen und Übertragungsmedien auf den Internet-Schichten 1 und 4 lassen sich beliebig austauschen.

**b)** Damit eine möglichst hohe Flexibilität bei den Übertragungsarten und Protokollen gewährleistet werden kann. Der Anwendung ist es gleich, über welche Arten der Übertragung sie mit einer Gegenstelle kommuniziert. Zudem wäre der Aufwand, die unteren Schichten zu ändern, um einiges höher und legacy code würde vielleicht nicht mehr funktionieren.

**c)** Nein, da die Übertragung über mehrere Schichten läuft und jede Schicht die Nutzerdaten der über ihr liegenden Schicht in eine PDU kapselt. Die logische Kommunikation, d.h. Protokoll arbeitet auf der jeweiligen Schicht, muss also mit der jeweiligen gegenüberliegenden Schicht kommunizieren. Ist das verwendete Protokoll unzuverlässig, kann dies zu Datenverlust führen.(z.B. beim Routing [fehlerhaftes Paket, Paket verlust, Paket umordnung])