Layer 4: Transport Layer

Der Layer 4 hat folgende 2 Aufgaben:

1) Bereitstellung von vielen Kommunikations-Endpunkten pro Host (damit verschiedene Anwendungen und auch verschiedene User gleichzeitig das Netzwerk benutzen können. Diese Endpunkte heißen **Ports** und sind durch einen `unsigned short` (in C/C++) realisiert. Pro Host gibt es daher ca. 65000 (2 hoch 16) Ports.
2) Empfangsbestätigung und gesicherter Verbindungsaufbau. Erst der Layer 4 stellt diese Möglichkeiten zur Verfügung.

Portnummern: Sowohl der Absender wie auch der Empfänger verwenden je einen Port. Während der Absender einfach einen freien, unbenutzten Port vom Betriebssystem anfordert, laufen am Server die Dienste unter festgelegten Portnummern: **Well known ports**. Diese sind von der Internetregulierungsbehörde in sogenannten RFCs (= Regelwerk des Internets) definiert und selbstverständlich über Google auffindbar. So findet man z.B.

http Port 80 (der WWW-Dienst)
smtp Port 25 (das Mail-Service)
ssh Port 22 (die secure shell)
…

Protokolle: TCP (Transmission Control Protocol), UDP (User Datagram Protocol)

UDP: Unter einem Datagramm versteht man Pakete ohne Empfangsbestätigung. Dieses Protokoll versendet also, wie schon alle bekannten Protokolle auf Layer 2 und 3, Pakete auf gut Glück. Dadurch sind auch Broadcasts und Multicasts möglich. Die wichtigsten Anwendungen sind das DHCP- und DNS-Service sowie Multimedia-Streaming, Voip (Voice over IP) und Netzwerk-Games. Es gibt auch herstellereigene Anwendungen zum Auffinden von kompatibler Hardware wie z.B. Apples Bonjour-Dienst (sucht z.B. Airplay-Wiedergabegeräte oder Airprint-Drucker) oder Microsofts vergleichbarer Dienst SSDP (Simple Service Discovery Protocol).

TCP: Erstellt eine „virtuelle Verbindung“. Zuerst muss die Verbindung aufgebaut werden (erfordert den Austausch von 3 Kontrollpaketen = 3way Handshake), dann wird bidirektional kommuniziert, am Ende wird die Verbindung wieder mittels 3way (Cisco sagt 4way) Handshake abgebaut. Jedes Datenpaket muss vom Empfänger bestätigt werden. Bleibt diese Bestätigung aus, wird das Paket so lange wiederholt gesendet, bis entweder eine Bestätigung eintrifft oder die Verbindung wegen des Zeitablaufs beendet wird. Zusätzlich erlaubt TCP auch die Geschwindigkeitsanpassung der Teilnehmer. TCP hat sich unter allen Umständen (schnelle und langsame Verbindungen) als hocheffizient herausgestellt. Es ist wegen der nötigen Host-Host Verbindung kein Broadcast oder Multicast möglich. Fast alle wichtigen Internetprotokolle benutzen heutzutage TCP!

Kommandos:

netstat –a (Windows und Unix) zeigt aufgebaute TCP-Verbindungen sowie sogenannte offene Ports an (d.h. ein Server betreut diese Ports und wartet auf Anfragen).

wireshark (Windows und Unix) erlaubt das Aufzeichnen des gesamten Netzwerk-Verkehrs zur Installation und Nutzung sind oft Administrationsrechte erforderlich.

3way Handshake: Die 3 Pakete des Verbindungsaufbaus von A zu B lassen sich leicht anhand der gesetzten Flagbits erkennen. Das sind Bits im TCP-Header, der jedes TCP-Segment einleitet. Die hierbei verwenden Bits sind SYN und ACK

1. Paket, A->B: SYN gesetzt: Es vereinbart die "Initial Sequence-Number" für A (ISN_A). A zählt alle seine gesendeten Bytes ab dieser Sequence-Number
2. Paket B->A: SYN+ACK gesetzt: Es vereinbart die "Initial Sequence-Number" für B (ISN_B). B zählt alle seine gesendeten Bytes ab dieser Sequence-Number. Zusätzlich bestätigt B auch den Empfang des 1. Pakets, indem er seine Acknowledgement-Number auf ISN_A +1 setzt.
3. Paket A->B: ACK gesetzt. A bestätigt den Empfang des 2. Pakets, indem er seine Acknowledgement-Number auf ISN_B+1 setzt.

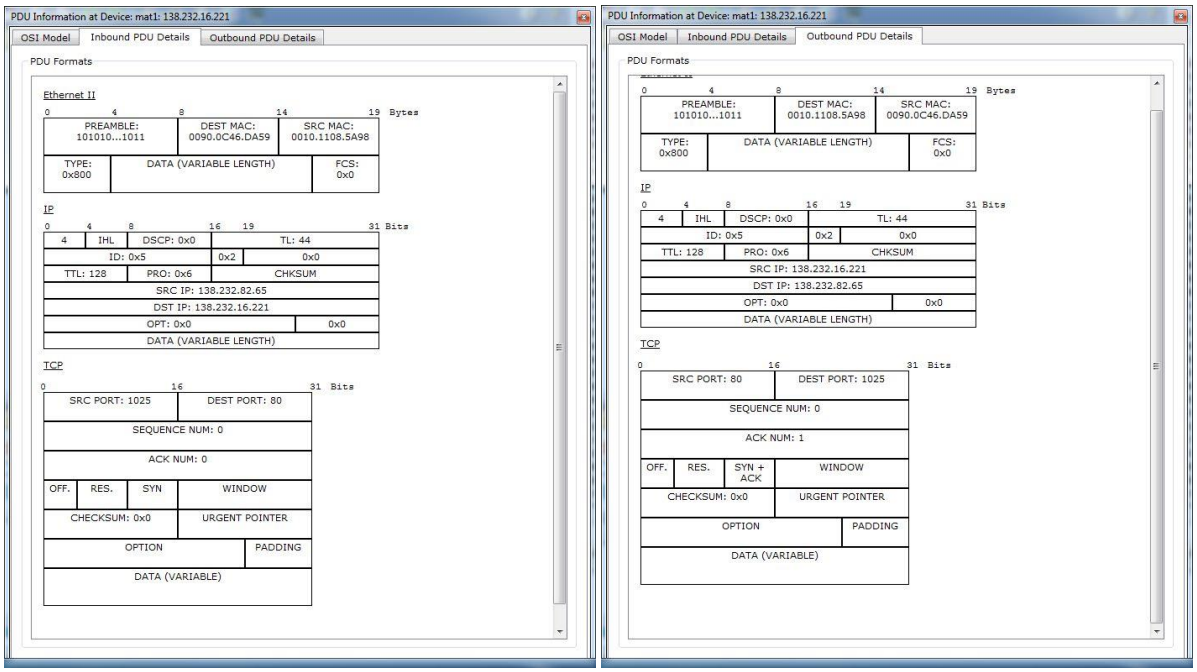
Anhand dieses Musters SYN, SYN+ACK, ACK kann man leicht einen Verbindungsaufbau erkennen. Beim Verbindungsabbau wird das FIN-Bit statt SYN verwendet.

Empfangs-Bestätigung: Ist in einem Paket das ACK-Bit gesetzt, so enthält es (u.U. neben Daten) auch eine Empfangsbestätigung. In der Acknowledgement-Number steht jene Bytenummer, die als nächstes erwartet wird, d.h. die kleinste Bytenummer, **die noch nicht empfangen wurde**. Meistens ist es die Nummer des zuletzt empfangenen Bytes + 1.

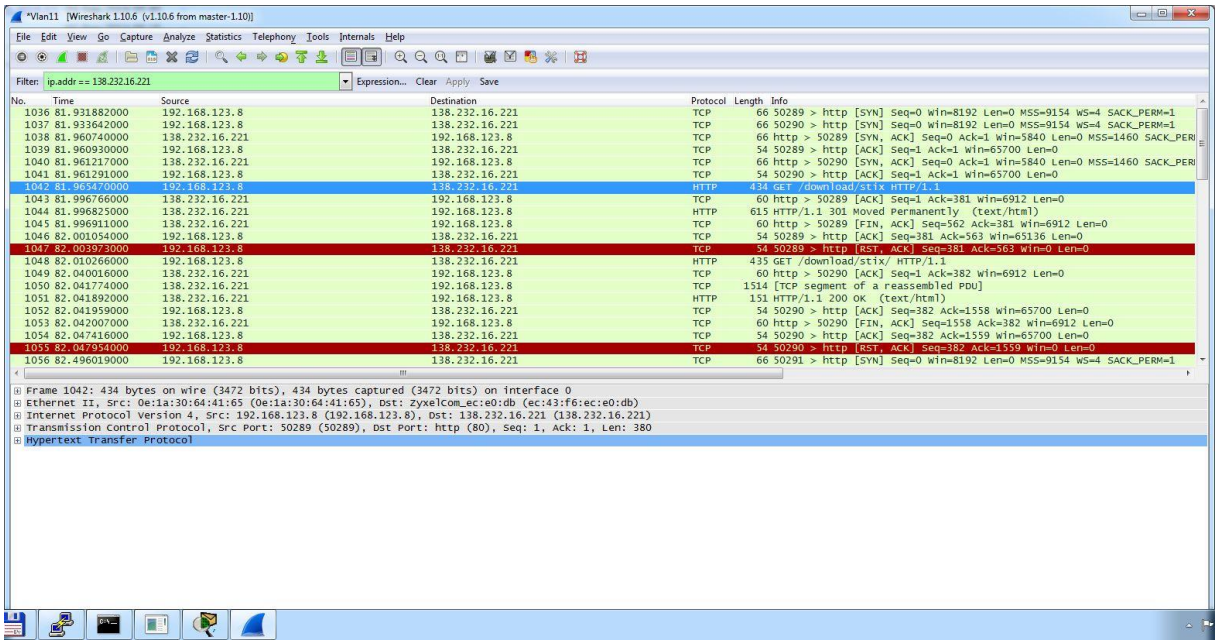
Window-Size: Gibt die Anzahl von Bytes an, die der Empfänger verarbeiten will (nicht kann!). Eine große Zahl bedeutet, dass der Sender schneller machen soll, eine 0, dass der Sender momentan nichts mehr senden soll. Beide Partner können ihr Gegenüber dadurch steuern.

Beispiel 5: 3-Way Handshake ("2 PCs.pkt")

Am PC straten wir den Webbrowser und laden die Hompage des Servers im Simulationsmodus. Zuerst erfolgt wie gewohnt die ARP-Sequenz. Dann geschieht mit dem 1. TCP-Paket der Handshake. Sehen wir uns dieses und die Antwort darauf an:



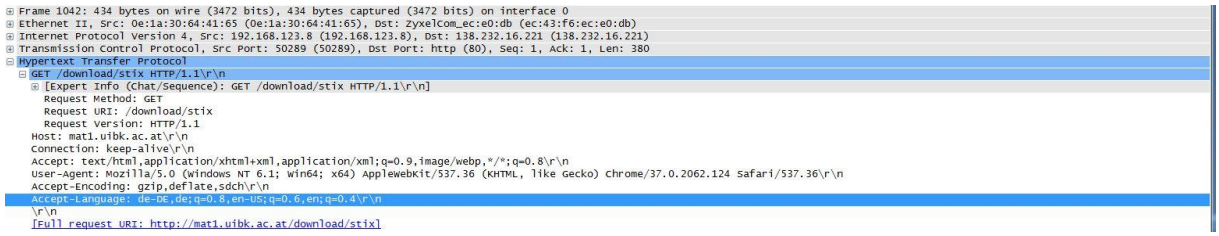
Im ersten Paket ist das SYS-Flag gesetzt und in der Anwort Syn+Ack. Beide Nodes beginnen ab 0 zu zählen (Sequence Num) und der Server bestätigt mit Ack Num = 1 die Sequence Num des PCs. Hier wie auch im 3. Paket des Handshakes sind noch keine echten Nutzdaten enthalten. Erst ab Paket Nr. 4 beginnt die eigentliche Kommunikation, die sich hier aber nur schwer verfolgen lässt. Belauschen wir mittels "Wireshark" einen echten WWW-Zugriff:



Ich habe auf meinem PC (192.168.123.8) mittels Chrome die URL: http://mat1.uibk.ac.at/download/stix

geladen. Man sieht zuerst den 3Way-Handshake, bei dem offenbar wegen Zeitüberschreitung das SYN-Paket 2 mal gesendet und bestätigt wurde (das erste Mal zu spät). Das darauffolgende Paket hat den Typ HTTP und ist daher schon ein Paket des Application-Layers, das hier im TCP-Segment steckt. Hier wird obige URL mittels GET angefordert. Man sieht auch dass dieses Paket genau 381 Bytes enthielt. Die darauf folgende Antwort des Servers hat die Länge 0 (enthält also keine Daten) sondern nur eine Empfangsbestätigung ACK NUM = 382, d.h. die Bytes 1-381 wurden empfangen und das 382. ist jenes, das als nächstes erwartet wird.

Warum ist der GET Request so lange? Sieht man sich diesen an, steht zusätzlich noch eine Menge von Infos über meinen Browser drin:



Manche Browser sind dabei so geschwätzig, dass sich aufgrund dieser Informationen ein eindeutiger Fingerabdruck berechnen lässt, der diesen Browser fast sicher identifizieren kann (wie ein DNA-Test). So kann z.B. Google Analytics meine Web-Aktivitäten tracken, egal unter welcher IP ich unterwegs bin (verschiedene Wlans, 3G Netze …). Die Seite **https://panopticlick.eff.org** testet, wie (wenig) anonym Sie im World Wide Web unterwegs sind. Das Gemeine daran ist: Je mehr Sie Ihren Browser absichern, desto eindeutiger werden Sie trackbar. Wer sich möglichst unauffällig im Web bewegen will, fällt gerade dadurch sofort auf. Dagegen hilft auch keine Verschlüsselung, weil Ihr Browser diese Daten jedem freiwillig überlässt und fast alle kommerziellen Webseiten einen dieser Personality-Tracker einbinden. Rufen Sie beliebige Webseiten auf, sehen Sie sich den Quellcode dieser Seiten an (geht in den meisten Browsern) und suchen Sie nach Tracking-Links (als Demonstration: http://www.krone.at und darin Google Analytics suchen!). Ebenso können Sie die Verlaufs-Funktion Ihres Browsers abchecken. Und damit haben Sie noch nicht einmal die Tracking-Möglichkeiten durch Cookies erfasst, die Ihnen bei fast jedem Seitenaufruf ungefragt untergejubelt werden.

PAT (Port Address Translation): kommt bei fast allen NAT-Routern zum Einsatz, um gleichzeitig mehrere Layer 4-Verbindungen (TCP und UDP) ins Internet zu routen. Beispiel: Man öffnet am eigenen PC (mit der IPv4-Adresse 192.168.1.100) einen Browser (oder einen neuen Tab darin) und gibt die URL http://mat1.uibk.ac.at/download/stix an:

1) Der Webbrowser fordert vom Betriebssystem einen freien TCP-Port an (etwa 4321).
2) Der Webbrowser ermittelt per DNS (Domain Name Service) die IPv4-Adresse des Servers mat1.uibk.ac.at: 138.232.16.221, Der WWW-Port ist 80 (well known)
3) Der PC versendet das 1. Paket des 3Way-Handshakes: 192.168.1.100:4321 -> 138.232.16.221:80

4) Der Router erkennt, dass es eine neue Verbindung ist und fordert einen freien TCP-Port von seinem Betriebssystem an, etwa 2222.
5) Der Router merkt sich in einer Datenbank 2222 -> 192.168.1.100:4321
6) Der Router öffnet seine Firewall von außen für 138.232.16.221:80 -> Port 2222
7) Der Router ersetzt im ausgehenden Paket die Absenderadresse (192.168.1.100) durch seine eigene offizielle IPv4-Adresse (etwa 80.80.80.80) (= NAT) und ersetzt die Absender-Portnummer (4321) durch seine eigene (2222) = PAT
8) Der Router sendet das Paket ins Internet: 80.80.80.80:2222 -> 138.232.16.221:80
9) Der Webserver antwortet mit dem 2. Paket des Handshakes: 138.232.16.221:80 -> 80.80.80.80:2222
10) Der Router lässt das Paket durch seine Firewall (siehe 6) und schlägt in seiner Datenbank nach: 2222 -> 192.168.1.100:4321
11) Der Router ersetzt die Zieladresse des Pakets durch 192.168.1.100, den Zielport durch 4321 und leitet das Paket ins Heimnetz weiter.
12) Der weitere Paketaustausch folgt obigem Schema. Beim Schließen der Verbindung löscht der Router den Eintrag aus der Datenbank und Firewall und gibt seinen Port 2222 ans Betriebssystem zurück.

PAT bei UDP funktioniert analog, nur dass es keinen Handshake für den Aufbau und Abbau der Verbindung gibt. Neue Verbindungen erkennt der Router nur durch das Nachsehen in seiner Datenbank (wenn sie dort nicht drinstehen), das Verbindungsende erfolgt meist per TimeOut. Jedes Paket startet den Timer neu.