ICT
§7 Netzwerk

Harald.Schil.ly
harald@schil.ly

050126

Netzwerk/Internet

Um zu verstehen wie Computer in einem Netzwerk wie dem Internet (oder kleiner, in einem Intranet) kommunizieren können, muss man mehrere Technologien verstehen. Diese werden in einem 7 Schichtenmodell (OSI) zusammengefasst. Vereinfacht gibt es das TCP/IP (DoD) Modell mit 4 Schichten:

1. **Netzwerk-Schicht** (*link layer*): Kommunikation zweier Geräte über eine Verbinung (Kabel, Glasfaser, Funk).
2. **Internet-Layer**: Vermittlung von Datenpaketen über mehrere Geräte hinweg - von einem Punkt im Netzwerk zu einem anderen.
3. **Transport-Layer**: Koordinierung des Versandes mehrerer Datenpakete: TCP, UDP
4. **Anwendungen** (application layer): Verschiedene Protokolle steuern, wie zwei oder mehrere Anwendungen untereinander mittels 1-3 miteinander kommunizieren (HTTP, FTP, SMTP, ...)

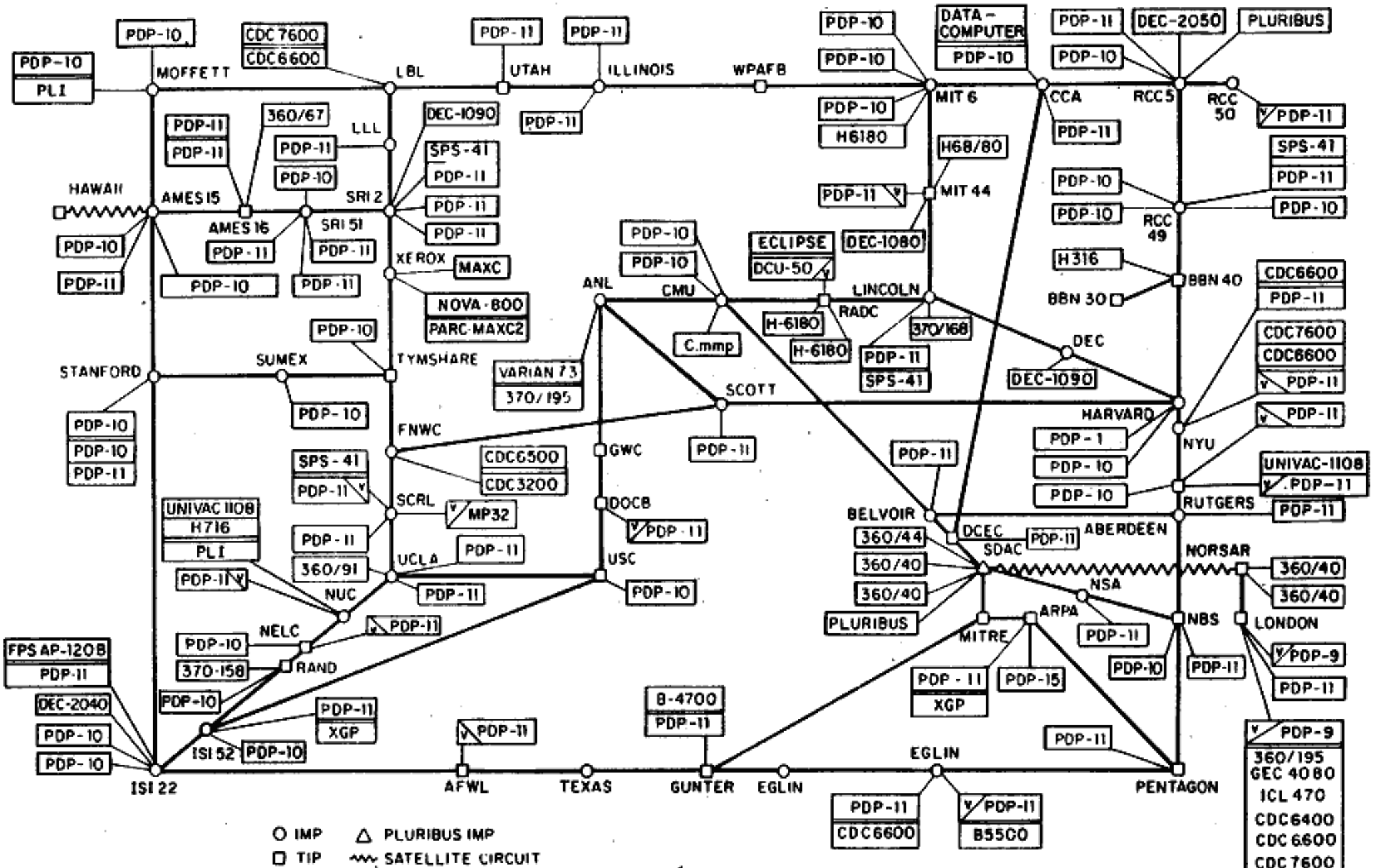
Netzwerk / RFC

Computernetzwerke haben eine lange Geschichte -> **ARPANET**.
Definiert wird deren Funktionsweise über die IETF. Diese gibt RFC ("request for comments") Dokumente heraus.
Besonderheit: frei erhältlich (open-source Gedanke), aufsteigend numeriert, einmal heraußen gibt es keine Veränderungen - nur neuere Dokumente die alte Ergänzen. -> auch als deb package 'doc-rfc' erhältlich.

- IRC Chat: 1459, EasterEgg: 2324, HTTP/1.1 2068
- RFC793 - Transmission Control Protocol (1981):

The **Transmission Control Protocol (TCP)** is intended for use as a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and in interconnected systems of such networks.
Lit: **Cerf, V.**, and R. Kahn, "A Protocol for Packet Network Intercommunication", IEEE Transactions on Communications, Vol. COM-22, No. 5, pp 637-648, May 1974.

ARPANET LOGICAL MAP, MARCH 1977

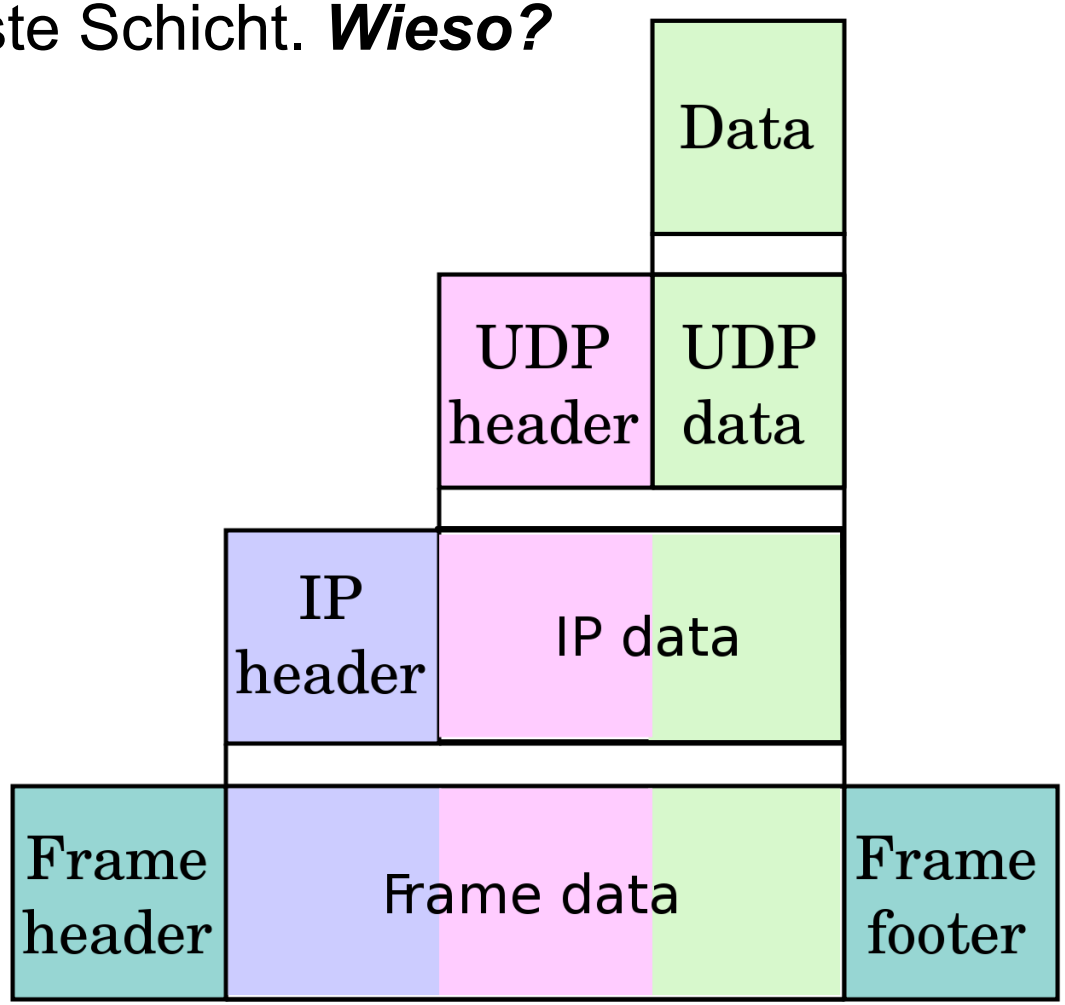


(PLEASE NOTE THAT WHILE THIS MAP SHOWS THE HOST POPULATION OF THE NETWORK ACCORDING TO THE BEST INFORMATION OBTAINABLE, NO CLAIM CAN BE MADE FOR ITS ACCURACY)

NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES

Netzwerk - Encapsulation

Diese Familie von Schichten verschickt Pakete, die ineinander verschachtelt sind. Je nach Typ des Geräts welches mit dem Paket arbeitet, wird es unterschiedlich weit aufgemacht - am häufigsten die äußerste Schicht. ***Wieso?***



Application

Transport

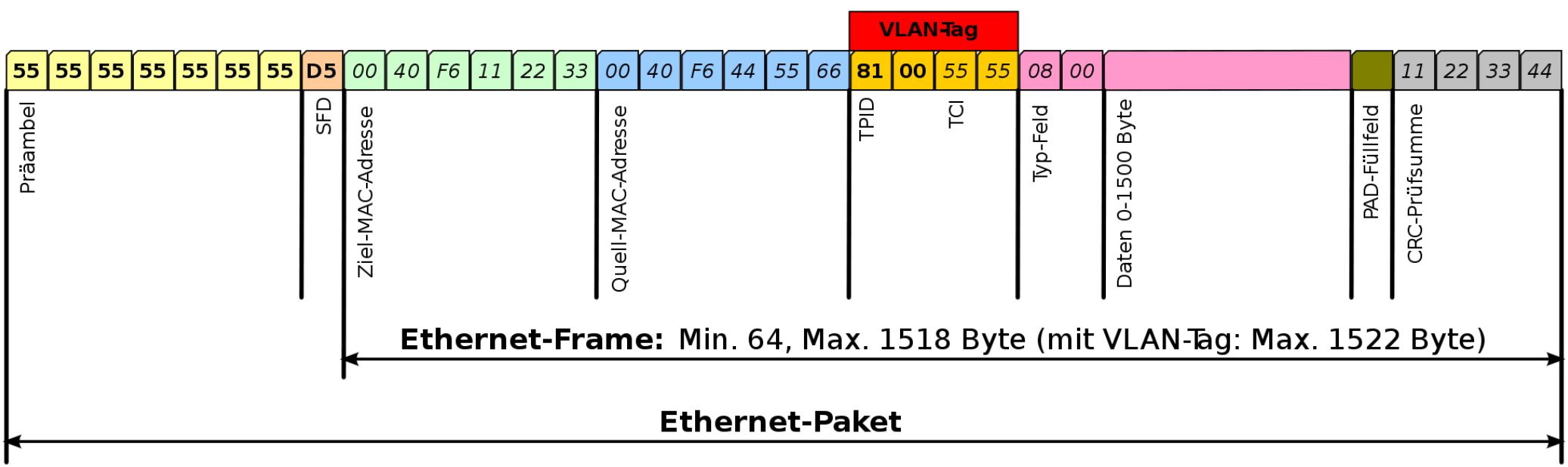
Internet

Link

Netzwerk - link layer

Dies ist die grundlegendste Schicht, sie regelt wie Netzwerkkarten mit anderen bzw. Routern interagieren. Jedes Gerät hat eine weltweit eindeutige Identifikation (Hardware Address "MAC" zB 00:04:ef:f1:..).

- Weit verbreitet ist das "Ethernet" (kompatibel zu W-Lan)
- Daten werden Prinzipiell nur in Paketen - "Frames" - verschickt.
- Aufbau:
  Header: Ziel+Quell MAC-Adresse > Daten > Prüfsumme



Netzwerk - IP Layer

Das **Internet Protokol (IP)** verschickt ebenfalls Datenpakete - *package* oder *datagram* genannt - welche die Daten des Network Layers sind! Die Adressen sind allerdings nicht die MAC Adressen, sondern IP-Adressen.

Es gibt zwei Typen: IPv4 (zB 131.130.70.17) oder IPv6 (zB 2001:0db8:85a3:...). IP Adressen müssen eindeutig sein, können sich aber für dasselbe Gerät ändern. IP-Adressen lassen sich gruppieren:
A, B und C-Klasse Netzwerk, Subnetting, CIDR TCP/IP (DoD) ModellNotation

Das Netzwerk ist ein Netz (Graph) wo in jedem Knoten ein "Router" sitzt. Die Hauptaufgabe des IP Layers ist es, Pakete zwischen weit voneinander entfernten Geräten über die Router zu verschicken. Es gibt keine zentrale Steuerung sondern jeder Router entscheidet nur anhand einer statischen oder dynamischen Tabelle, diverser Algorithmen und seiner "Erfahrung". In der Praxis sind viele Knoten vom selben Anbieter und dieser steuert sie zentral und tauscht mit anderen Pakete aus.

Netzwerk - IP Protocol

Ein IPv4 Datagram besteht aus einer Start- und Ziel IP-Adresse, einigen Optionen und einem TTL Feld.

TTL steht für "time to live" und limitiert die Anzahl von Zwischenstellen die ein Paket passieren darf. Es wird verhindert, dass es ewig im Netzwerk herumirrt. Üblicher Startwert zwischen 32 bis 128, wird bei jedem Knoten um 1 vermindert. Geringere TTL Werte werden für Tracerouting verwendet, da ein überschreiten des TTL Limits als Fehler (ICMP Message) rückgemeldet wird.

Besondere IP Adressen:

- 10.x.x.x und 192.168.x.x für lokales Netzwerk
- 0.0.0.0 und 255.255.255.255 (broadcast) spezielle Zwecke

Netzwerk - ARP

Für das Verschicken von einem Netzwerk-Daten-Frame über IPv4 muss man die MAC Adresse des Nachbar-Rechner/Router wissen. Wenn man nur die IP Adresse weiß, muss die Hardware Adresse ermittelt und zwischengespeichert werden. Das Übernimmt das ARP (Address Resolution Protocol). Es assoziiert IP-Adressen mit MAC Adressen.

```
$ arp
Address                  HWtype  HWaddress           Flags
Mask            Iface
damokles.cc.univie.ac.a  ether   00:1f:9e:ae:97:00   C   eth0
seppukuneu.mat.univie.a  ether   00:16:3e:00:16:11   C   eth0
goedel.mat.univie.ac.at  ether   00:14:38:c6:e3:a1   C   eth0
alexandria.mat.univie.a  ether   00:60:16:06:8c:62   C   eth0
broadcast.mat.univie.ac  ether   00:13:72:a4:7c:91   C   eth0
famulus.mat.univie.ac.a  ether   00:13:72:a4:72:25   C   eth0
```

IPv6 verwendet etwas ähnliches: NDP

Netzwerk Transport - ICMP

ICMP steht für "Internet Control Message Protocol".

Es ist sehr einfach und dient nur dazu, Steuerbefehle zwischen Routern auszutauschen. Es werden keine Daten übertragen.

Am bekanntesten ist der "Ping", welcher benützt wird um zu testen ob ein anderer Rechner über eine IP-Adresse erreichbar ist. (Die andere Rechner muss den Ping empfangen und von sich aus ein "Pong" Signal zurückschicken)

```
$ ping <ip-address>
PING 131.130.70.17 (131.130.70.17) 56(84) bytes of data.
64 bytes from 131.130.70.17: icmp_seq=1 ttl=125 time=1.00 ms
64 bytes from 131.130.70.17: icmp_seq=2 ttl=125 time=0.975 ms
64 bytes from 131.130.70.17: icmp_seq=3 ttl=125 time=0.985 ms
```

Netzwerk - Traceroute

Tracerouting zeigt die Liste von Knoten an, die ein Datenpakt durch das Internet nimmt. Es wird eine Reihe von Paketen mit fortlaufender TTL Zahl (1 bis 30 oder mehr) abgesetzt. Knoten bei denen die TTL Zahl 0 wird, verwefen das Paket und senden ein ICMP vom Typ 11 zurück ("TTL exceeded").

Die Laufzeiten der Pakete sowie die Liste dieser Fehler wird verwendet, um Informationen über den gewählten Pfad zu sammeln.

Link: Traceroute (wikipedia, en)

Netzwerk - Traceroute / 2

$ mtr -t www.mexicocity.gob.mx

```
 Host                                          Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. pan.cc.univie.ac.at                         2.9%    34    1.2   2.8   1.1  10.5   2.7
 2. ares.cc.univie.ac.at                        0.0%    34    1.2   6.0   1.1 131.9  22.3
 3. vlan1501.wien2.aco.net                      0.0%    34    5.9   2.1   1.1   7.0   1.6
 4. vlan72.wien21.aco.net                       0.0%    34    1.4   2.7   1.3  23.4   3.9
 5. 212.73.203.17                               0.0%    34    2.5  31.3   1.4 187.0  59.8
 6. ae-11-11.car1.Vienna1.Level3.net            0.0%    34   88.2  20.4   1.4 214.3  46.4
 7. ae-6-6.ebr1.Frankfurt1.Level3.net           0.0%    34   13.0  14.1  13.0  22.6   2.0
 8. ae-91-91.csw4.Frankfurt1.Level3.net         0.0%    33   17.0  20.7  13.2  61.9   8.7
 9. ae-92-92.ebr2.Frankfurt1.Level3.net         0.0%    33   14.5  15.2  13.3  30.9   3.2
10. ae-42-42.ebr2.Washington1.Level3.net        0.0%    33  131.0 104.7 102.5 131.0   5.1
11. ae-62-62.csw1.Washington1.Level3.net        0.0%    33  103.3 107.2 101.8 115.7   4.4
12. ae-1-69.edge1.Washington4.Level3.net        0.0%    33  136.3 106.4 102.1 142.2   9.0
13. 4.68.62.34                                  0.0%    33  106.0 108.4 105.0 168.3  11.0
14. cr2.wswdc.ip.att.net                        0.0%    33  169.6 170.8 168.9 178.4   2.1
15. cr1.attga.ip.att.net                        3.0%    33  182.3 183.4 180.6 205.7   4.6
16. cr2.dlstx.ip.att.net                        0.0%    33  170.0 171.4 168.7 207.9   6.7
17. cr2.la2ca.ip.att.net                        0.0%    33  182.2 171.5 170.2 182.2   2.1
18. 12.123.30.137                               0.0%    33  176.2 171.1 168.4 188.0   4.2
19. 12.89.4.6                                   0.0%    33  169.5 171.6 169.1 186.7   3.4
20. ???
21. ns1.mexicocity.gob.mx                       0.0%    33  226.1 239.9 218.7 299.2  19.3
```

In der Ping-Zeit kommt auch die Lichtgeschwindigkeit ins Spiel!

Netzwerk - Traceroute / 3

$ mtr -t www.net.tw

```
 Host                                          Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 192.168.0.1                                 0.0%    18  111.2 122.7   2.1 243.4  89.1
 2. vie-086-059-083-001.dsl.sil.at              0.0%    18   62.3 141.1  17.6 287.4  92.1
 3. Ge2-3-c1.ix.sil.at                          0.0%    18   17.5 135.7  17.5 366.9 100.6
 4. peer1.fra1.flagtel.com                     33.3%    18   34.5 166.9  34.5 398.4 124.8
 5. so-1-1-1.0.pjr02.ldn001.flagtel.com        38.9%    18  333.9 515.3 333.5 843.0 195.1
 6. so-1-0-0.0.pjr01.nyc007.flagtel.com        41.2%    18  771.5 602.9 347.3 848.0 165.9
 7. 85.95.26.22                                35.3%    18  698.3 651.5 347.3 819.0 179.4
 8. ge-5-0-0.0.cjr02.lax002.flagtel.com        41.2%    18  314.1 362.6 208.6 503.3 100.5
 9. so-0-1-0.0.cjr04.tok002.flagtel.com        35.3%    18  551.5 397.5 308.3 551.5  89.3
    so-5-2-0.0.cjr04.tok002.flagtel.com
10. so-0-3-0.0.pjr02.wad001.flagtel.com        41.2%    17  508.1 509.0 337.2 661.9 115.4
11. so-5-0-0.0.ejr01.tpe001.flagtel.com        35.3%    17  446.7 577.2 351.6 843.5 188.6
12. 62.216.145.182                             35.3%    17  358.3 439.7 330.7 582.3  80.6
13. 202.133.224.84                             35.3%    17  593.8 546.6 380.0 699.0 114.9
14. 202-153-173-34-static.unigate.net.tw       35.3%    17  506.4 471.7 329.6 593.5 107.3
```

Portnummern

Wenn Daten über eine Netzwerkschnittstelle hereinkommen, muss das Betriebssystem sicherstellen, dass nur die richtige Anwendung (laufendes Programm) die Daten erhält. Ansonsten könnte ein anderer Prozess sensitive Daten auslesen! Hierfür wird eine **"Port"-Nummer** verwendet.

Prozesse müssen, bevor sie Daten senden oder empfangen können, beim Betriebssystem eine Portnummer reservieren. Benutzer-Anwendungen können dies erst ab Portnummer 1024.

Wichtige Portnummern:
<1024 nur für Administratoren und Server, fix vergeben:
- 80: Webserver; 22: SSH; 21: FTP; ...

>=1024 für Anwendungssoftware, frei verfügbar wenn nicht belegt.
- 8080: lokaler Webserver, Internet-Telefonie, ...

Linux: **/etc/protocols** ... rein "technisch" kann aber jedes Programm jeden Port verwenden.

Netzwerk Transport - UDP

UDP (User Datagram Protocol) ist ein sehr einfaches Protokoll, welches Datenpakete abkapselt, mit einer Prüfsumme versieht, und mit einer Portnummer addressiert.

Es gibt alle "Probleme" des IP auch hier:

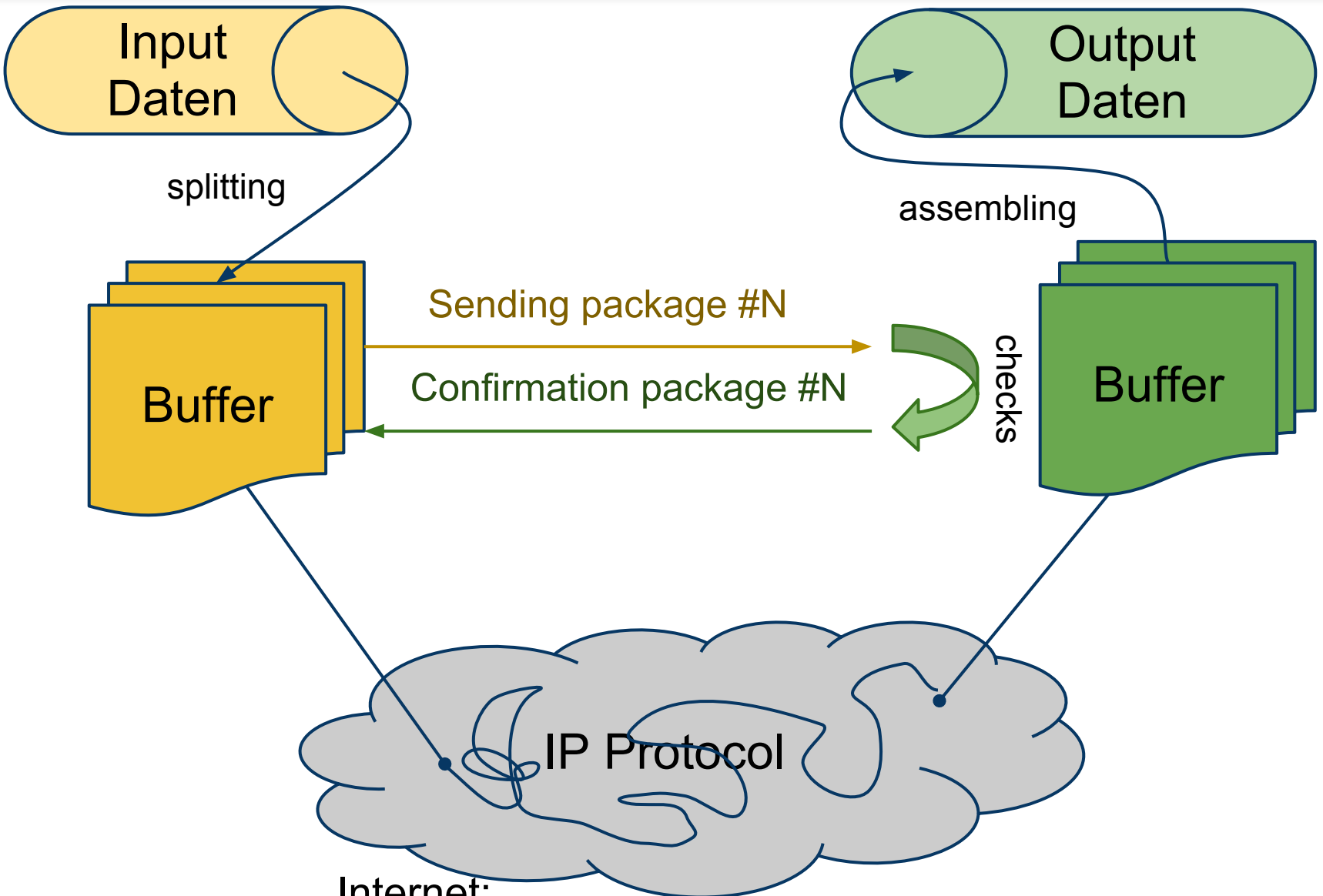
- Es wird keine dauerhafte Verbindung aufgebaut.
- Der Empfang wird nicht garantiert.
- Die Reihenfolge kann sich ändern.
- Die Applikation muss die zusammengefügten Daten selbst prüfen bzw. Teile erneut anfordern.
  - Es wird kein Check für Duplikate gemacht.

Netzwerk Transport - TCP

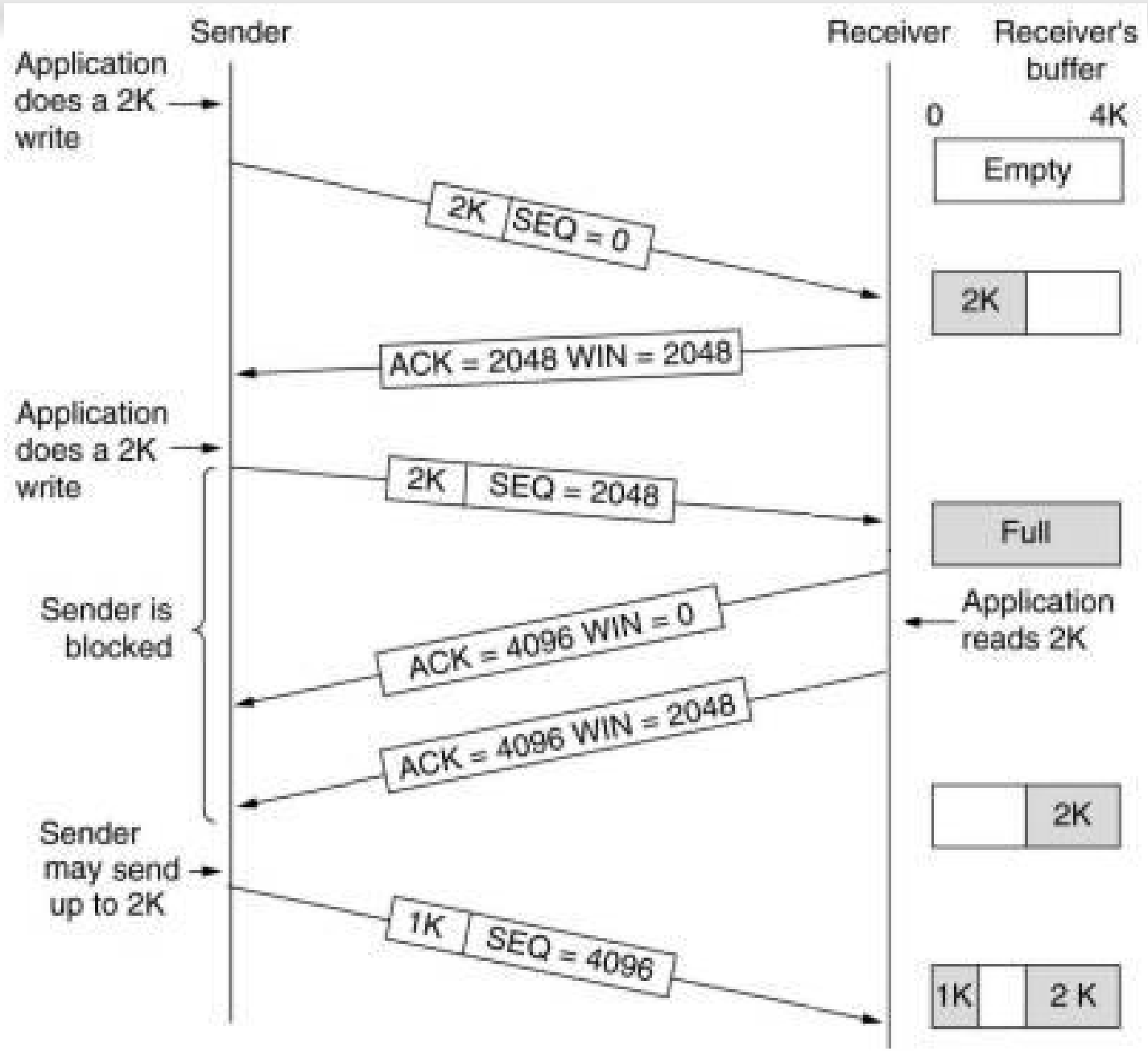
TCP (Transmission Control Protocol) ist das am häufigsten verwendete Protokoll und Herzstück des modernen Internets. So wie bei UDP gibt es Portnummern. Zusätzlich wird eine permanente Verbindung ("Handshake") aufgebaut, die auch dann besteht wenn keine Daten versendet werden (Timeout ein paar Minuten).
Der Sender teilt die Gesamtmenge der zu verschickenden Daten in Pakete auf, nummeriert sie und versieht sie jeweils mit Prüfsummen. Die Empfängerseite kontrolliert ob alle Teilstücke korrekt ankommen, fehlende Teile werden erneut angefordert und es wird geregelt wie schnell der Sender die Daten verschicken darf.
Dadurch können Applikationen robust Daten austauschen ohne sich mit diesen Details beschäftigen zu müssen.
Promineter Verwendungszweck: HTTP im Web-Browser.

TCP-Header enthält u.a.: Quell+Ziel Portnummer, Sequenznummer, Teilstückgröße und die Prüfsumme.

Netzwerk Transport - TCP /2



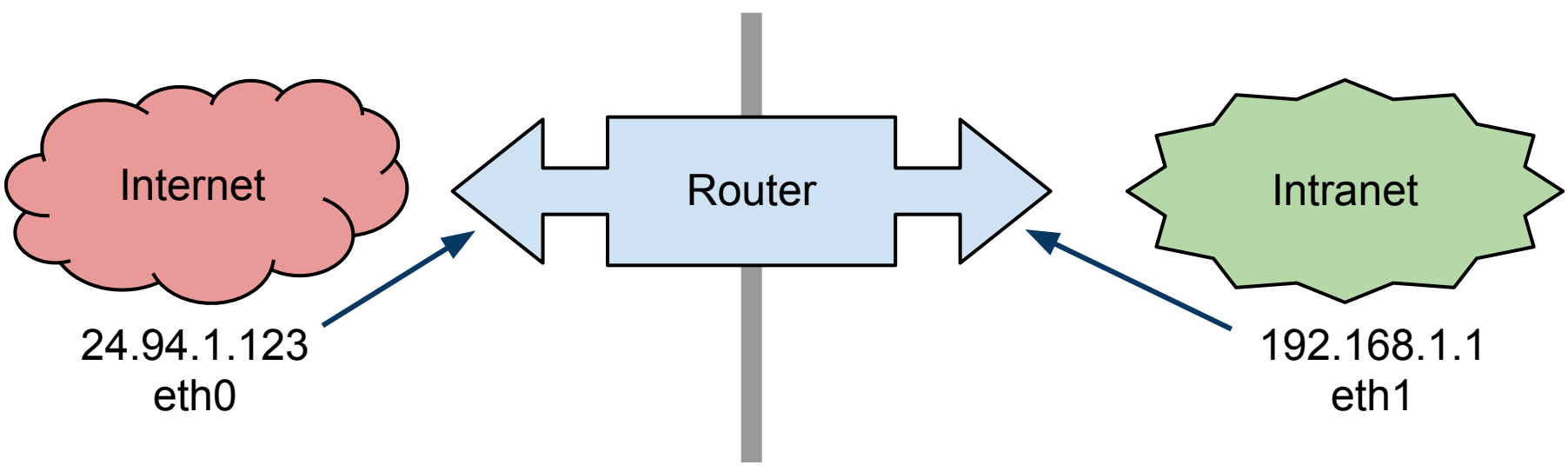
Netzwerk Transport - TCP /3



Netzwerk - Router

Eine Recheneinheit, die mit mehreren anderen über ein Netzwerk verbunden ist heißt Router. Agiert er anhand einer statischen Tabelle, so kann dieser zB ein Gateway zum Internet sein.
Konfiguration in Linux mit dem Befehl '`route`' und '`iptables`':

```
$ route -n
Kernel routing table
Destination     Gateway         Genmask         Flags MSS    Window Use Iface
24.94.1.0       *               255.255.255.0   U     1500   0        15  eth0
192.168.1.0     *               255.255.255.0   U     1500   0         0  eth1
127.0.0.0       *               255.0.0.0       U     3584   0         2 lo
default         24.94.1.123     *               UG    1500   0        72  eth0
```



Netzwerk - IP - Router / 2

... und '`iptables`':
Idee: 3 basic chains: ***input, output, forward*** - man kann zusätzliche erzeugen.
Liste von Regeln klärt, was in den chains weiterbefördert wird:
`'-A' append, '-I' insert, '-R' replace, '-D' delete`
Am Ende ist ein Target: Accept, Drop (Reject), Redirect, ... oder eigene Chain.
Konfiguration: zuerst Bisheriges löschen (!), dann Ketten aufbauen.

Bestehende Verbindungen akzeptieren (wichtig)
```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Webserver auf Port 80 (von überall akzeptieren)
```
iptables -A INPUT -p TCP -s 0/0 --destination-port 80 -j ACCEPT
```

Regeln nur im Kernel gespeichert, es gibt Skripte zum Speichern und automatischem neu laden: `iptables-save/-restore`
www: iptables tutorial, man iptables, Konfigurationstools (firestarter), Skriptgeneratoren, ...

Netzwerk - DHCP

Jedem Teilnehmer im Netzwerk muss eine (global, oder im lokalen Intranet Netz) eindeutige IP Adresse zugewiesen werden. Das kann man statisch vornehmen (`ifconfig` für `eth0`) oder automatisch über einen Server im Netzwerk - dem DHCP-Server.

Dieser reagiert auf bestimmte Broadcast Pakete an die IP 255.255.255.255, Port 67, UDP Protokoll, welche um eine IP Adresse + Konfiguration bitten.

Er antwortet mit einer neuen gültigen IP Adresse und teilt dem Rechner zB auch den Router mit, über den man ins Internet kommt, einem Nameserver, DNS Server oder eine Proxy-Konfiguration.
Die vergebene IP Adresse kann zeitlich beschränkt sein: dann muss der Client immer wieder um eine neue bitten.

Lit: DHCP

Netzwerk - Protokoll

Damit nun zwei Programme über das Netzwerk kommunizieren können, müssen sie sich auf eine gemeinsame **Sprache** einigen - das nennt man **Protokoll** (ganz oben im Application Layer).
Beispiele:
HTTP (Webserver, RFC 1945), POP (E-Mail Postfach), SMPT (E-Mail versenden), FTP (Dateiübertragung), ...

Der Server bekommt Anfragen (Verb + Daten) schickt eine Antwort.

Generell unterscheidet man auch, ob der Server einen "Status" hat, ob w
Befehle einen Unterschied machen ("Idempotenz"), es gibt auch Authentifizierungen, ...

Link: 28c3: Science of Insecurity über die Probleme dahinter…

Netzwerk - Protokoll - HTTP - 1

Beispiel HTTP:

- Anfrage: Verb: **`GET`** + Pfad für Datei + Protokollversion

```
GET /index.html HTTP/1.1
```

- Antwort: Statusmeldung + Typ und Inhalt der Datei (Webseite, in HTML geschrieben, oder Binärdatei wenn es zB ein Bild ist)

- Verb **`POST`**: zum senden von Daten (Formularfelder)

- Andere: **`HEAD`** (nur Header), **`DELETE`**, **`PUT`** und **`PATCH`**. Wird für ReST Services (Representational State Transfer, Stateless) verwendet - das ist eine Anwendung neben dem Transfer von Webseiten.

Netzwerk - Protokoll - HTTP - 2

Beispiel HTTP: Ein sehr einfacher Webserver:

```
#!/usr/bin/env python
"""very basic http webserver"""
from BaseHTTPServer import BaseHTTPRequestHandler, HTTPServer
import time
class HttpHandler(BaseHTTPRequestHandler):
 def do_GET(self):
  try:
   self.send_response(200)
   self.send_header('Content-type', 'text/html')
   self.end_headers()
   self.wfile.write("%s:%s requested " % self.client_address)
   self.wfile.write("%s at %s <br>" % (self.path, time.asctime()))
   self.wfile.write(open('.'+self.path).read())
  except:
   self.send_error(404, '%s does not exist'%self.path)

# start server on port 8080
server = HTTPServer(('', 8080), HttpHandler)
server.serve_forever()
```

Netzwerk - Protokoll - HTTP - 3

Beispiel HTTP: Ein sehr einfacher Webbrowser mit Telnet:

```
$ telnet localhost 8080
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET /index.html HTTP/1.0

HTTP/1.0 200 OK
Server: BaseHTTP/0.3 Python/2.6.4
Date: Wed, 24 Mar 2010 21:49:34 GMT
Content-type: text/html

<h1>Hello 127.0.0.1:52664</h1>you requested /index.
html<br/>time: Wed Mar 24 22:49:34 2010

Connection closed by foreign host.
```

Telnet schickt Anfrage
2x Return drücken

HTML Header

HTML Content

Netzwerk - DNS Server

DNS - Domain Name Server. Das ist ein Dienst, der aussagekräftige Namen für IP Adressen anbietet. Es gibt einen zentralen DNS Server, einige die von ihm die Daten beziehen und jeder ISP wiederrum bezieht von diesen die Einträge. (`dig <name> +trace`)

```
$ dig www.derstandard.at +short
194.116.243.20
$ dig -x 194.116.243.20 +short #reverse lookup
www.derstandard.at.
```

Ohne "+short" Option viel mehr Output. Andere Optionen: "+noall" "+answer", ... Es kann nach Typ gefiltert werden (nur Mailserver, ...)
DNS Einträge werden Zwischengespeichert, daher wirken sich Änderungen am Hauptserver erst nach einiger Zeit (Stunden) aus.
Link: Dig Howto
Ähnlich, aber einfacher: $ **host**

Netzwerk - lsof

`lsof`: erlaubt nicht nur das Monitoring von geöffneten Dateien, sondern auch von geöffneten Netzwerkverbindungen.

```
$ lsof -r 2 -p <PID> -i -a
```

- `-i` ... Interface
- `-a` ... logisches "und" für die Optionen.
- `-r 2` ... repeat-Modus alle 2 Seks, `Strg-C` zum Beenden
- `-p` ... spezifiziere PID einer Anwendung
- `-u` ... beobachte was ein bestimmter User macht:
  - `lsof -r 2 -u username -i -a`

Netzwerk / lsof -i

offene Verbindungen:

```
$ lsof -P -i -n
COMMAND     PID    USER   FD   TYPE  DEVICE SIZE NODE NAME
ssh         991 schilly    3u  IPv4 1674908       TCP 131.130.16.104:47782->131.130.16.129:389
(CLOSE_WAIT)
ssh         991 schilly    6u  IPv4 1674918       TCP 131.130.16.104:48372->128.208.160.197:22
(ESTABLISHED)
chrome      994 schilly   80u  IPv4 1733600       TCP 131.130.16.104:40423->69.63.186.12:80
(ESTABLISHED)
...
ksmserver  3558 schilly   21u  IPv4   11945       TCP 131.130.16.104:55995->131.130.16.129:389
(CLOSE_WAIT)
kwin       3560 schilly   11u  IPv4   11198       TCP 131.130.16.104:55911->131.130.16.129:389
(CLOSE_WAIT)
kwrited    3569 schilly   12u  IPv4   11306       TCP 131.130.16.104:55916->131.130.16.129:389
(CLOSE_WAIT)
knotify4   3578 schilly   12u  IPv4   13023       TCP 131.130.16.104:56098->131.130.16.129:389
(CLOSE_WAIT)
nepomukse  3579 schilly   11u  IPv4   11490       TCP 131.130.16.104:55930->131.130.16.129:389
(CLOSE_WAIT)
plasma     3582 schilly   14u  IPv4   11362       TCP 131.130.16.104:55920->131.130.16.129:389
(CLOSE_WAIT)
kaccess    3617 schilly   15u  IPv4   12091       TCP 131.130.16.104:56010->131.130.16.129:389
(CLOSE_WAIT)
kxkb       3619 schilly   16u  IPv4   12086       TCP 131.130.16.104:56009->131.130.16.129:389
(CLOSE_WAIT)
krunner    3629 schilly   21u  IPv4   12790       TCP 131.130.16.104:56078->131.130.16.129:389
(CLOSE_WAIT)
...
```

Netzwerk / ss

offene Sockets:

```
$ ss -p | column -t
State       Recv-Q  Send-Q  Local                  Address:Port          Peer
CLOSE-WAIT  1       0       131.130.16.104:56098   131.130.16.129:ldap   users:(("knotify4",3578,12)
CLOSE-WAIT  1       0       131.130.16.104:55916   131.130.16.129:ldap   users:(("kwrited",3569,12))
ESTAB       0       0       131.130.16.104:36872   74.125.87.113:www     users:(("chrome",994,107))
CLOSE-WAIT  1       0       131.130.16.104:54969   131.130.16.129:ldap   users:(("kio_file",10220,20
CLOSE-WAIT  1       0       131.130.16.104:56033   131.130.16.129:ldap   users:(("bash",3642,3))
CLOSE-WAIT  1       0       131.130.16.104:55911   131.130.16.129:ldap   users:(("kwin",3560,11))
CLOSE-WAIT  1       0       131.130.16.104:56768   131.130.16.129:ldap
ESTAB       0       0       131.130.16.104:54171   74.125.87.101:www     users:(("chrome",994,116))
CLOSE-WAIT  1       0       131.130.16.104:56009   131.130.16.129:ldap   users:(("kxkb",3619,16))
ESTAB       0       0       131.130.16.104:36871   74.125.87.113:www     users:(("chrome",994,94))
CLOSE-WAIT  1       0       131.130.16.104:56047   131.130.16.129:ldap   users:(("bash",3646,3))
CLOSE-WAIT  1       0       131.130.16.104:56052   131.130.16.129:ldap   users:(("bash",3661,3))
ESTAB       0       0       131.130.16.104:56892   69.63.184.142:www     users:(("chrome",994,80))
CLOSE-WAIT  1       0       131.130.16.104:55475   131.130.16.140:ldap
CLOSE-WAIT  1       0       131.130.16.104:38009   74.125.87.105:www     users:(("chrome",994,61))
CLOSE-WAIT  1       0       131.130.16.104:56027   131.130.16.129:ldap   users:(("bash",3641,3))
CLOSE-WAIT  1       0       131.130.16.104:55758   131.130.16.129:ldap
ESTAB       0       0       131.130.16.104:50335   74.125.87.120:www     users:(("chrome",994,99))
CLOSE-WAIT  1       0       131.130.16.104:56053   131.130.16.129:ldap   users:(("konsole",3634,22))
...
```

column -t dient nur der Verschönerung

Netzwerk Statistik / ss -s

Sehr einfache Statistik über die offnenen Verbindungen:

```
$ ss -s
Total: 602 (kernel 605)
TCP:   68 (estab 16, closed 6, [...]), ports 0

Transport Total     IP        IPv6
*         605       -         -
RAW       0         0         0
UDP       12        11        1
TCP       62        56        6
INET      74        67        7
FRAG      0         0         0
```

gut kombinierbar mit watch

TCP Tuning / sysctl

TCP-Tuning: Das TCP Protokoll verwendet einen Sende- u. Empfangsbuffer. Ist dieser zu klein, kann nur mit reduzierter Geschwindigkeit übertragen werden. Hierfür wird das **BDP** (Bandwidth-Delay-Product) als Richtwert berechnet.

- **RTT** = "Round-Trip-Time" = Laufzeit ab dem Senden eines Pakets bis eine "ACK" Antwort zurückkommt.
- **link bandwidth** = Geschwindigkeit, mit der Datenpakete im Netzwerk unterwegs sein können.

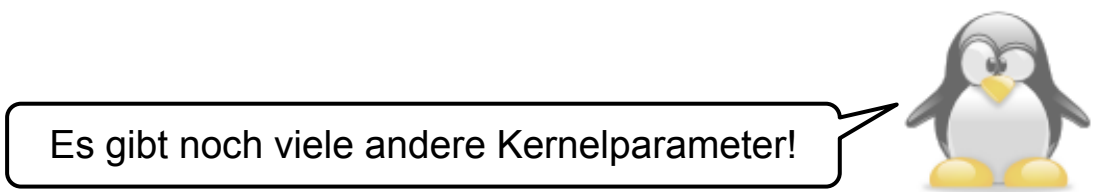
BDP = RTT * link bandwith

Beispiel: 10 megabit/s * 500 ms = 640 kilobytes

TCP Tuning / sysctl / 2

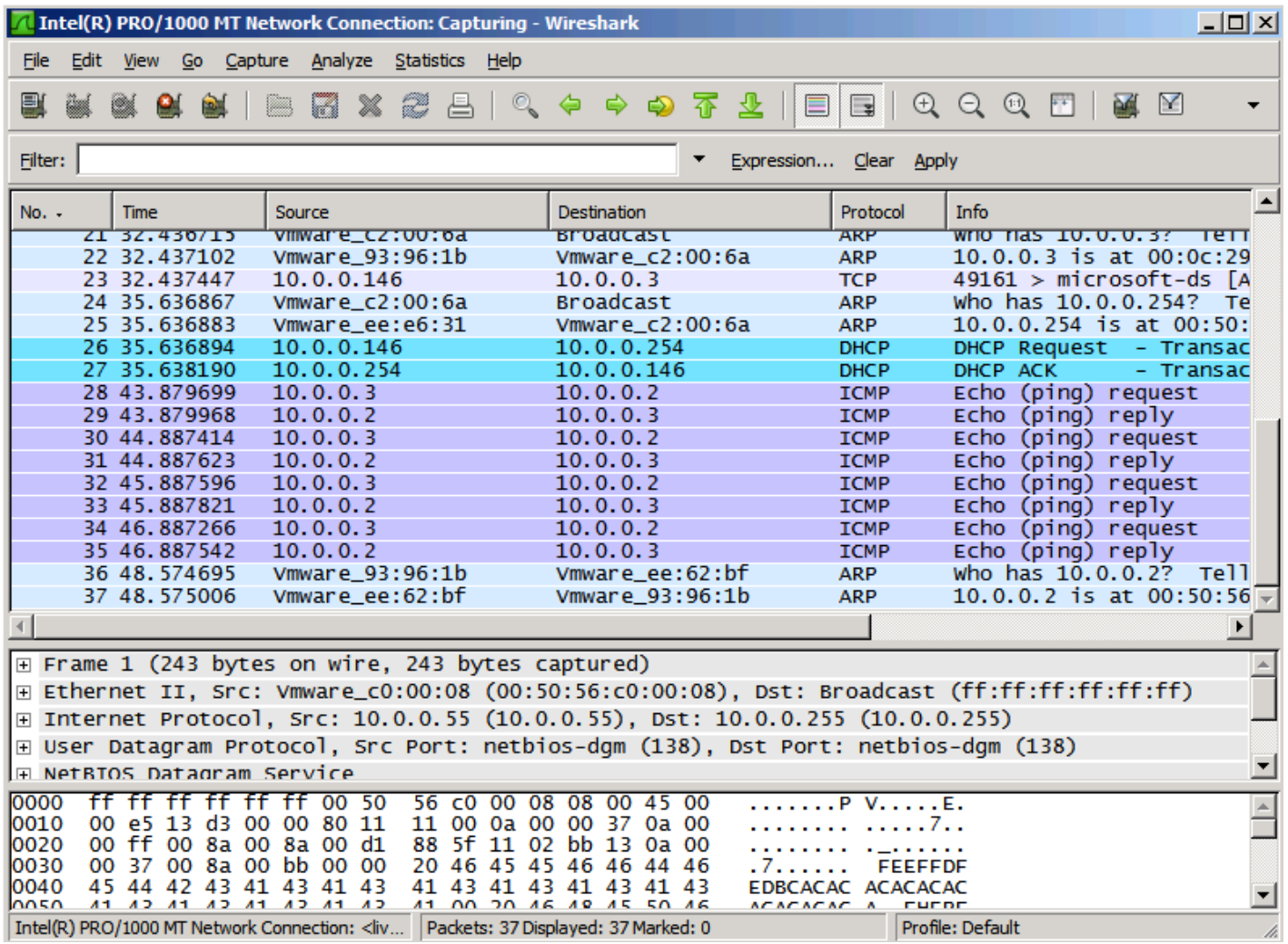
Die Parameter für die TCP Verbindung sind Kernelparameter. Diese befinden sich in `/proc/sys/net/ipv4/*` und können mittels `cat` bzw. über `sysctl` ausgelesen werden. Werte können (als root) mittels `echo "wert" > file` bzw. ebenfalls über `sysctl -w` gesetzt werden.

Alle Einstellungen sind nur temporär bis zum nächsten Reboot. Permanent werden sie in `/etc/sysctl.conf` festgelegt. Änderungen werden (System-V init Skript vorausgesetzt) mit `invoke-rc.d procps start` aktiviert.



Netzwerk Monitoring /w Wireshark

Besseres Monitoring kann man mit diversen Utilites machen. Ein sehr umfangreiches ist Wireshark. (`sudo apt-get install wireshark`)



- Menü:
  Statistik, Analyse, Graphen, ...
- Filter
  zB "tcp"
- Liste der Pakete
- Erkennen des Protokolls
- Rekonstruktion des Datenstroms bei TCP

Netzwerk Monitoring Utilities

Das deb Paket **`netdiag`** enthält weitere interessante Tools, jeweils für Vollbild-Kommandozeile:

- trafshow - zeigt Verbindungstraffic an
- netload - wie stark ein Netzwerkgerät verwendet wird
- checkint - welche sind aktiv
- statnet - daemon + client, sammelt Statistiken
- tcpspray - durchschnittlicher Datendurchsatz pro Verbindung, braucht Server
- strobe - sucht offene TCP ports
- netwatch - sammelt Statistik über alle IP Pakete (Quelle+Ziel)

Weitere:

- netstat - zeigt offene Verbindungen an
- atop - Ähnlich wie "top" (braucht kernel patch für Netzwerk)
- ntop - Statistiken im Browser als Webseite
- iptraf - Zeigt aktive Verbindungen an, Transferstatistik, TCP
- tcpdump - TCP Infos speichern, später analysieren
- speedometer - Downloadspeed Messgerät für Datei/Netzwerk

Links

http://www.faqs.org/docs/linux_network/x-087-2-intro.outlook.html

Copyright & Lizenz

Copyright:

Mag. Harald Schilly <harald.schilly@univie.ac.at>
© 2012, Wien, Österreich.

Lizenz:

Creative Commons **CC BY-NC-SA**
"Namensnennung-Keine kommerzielle Nutzung-Weitergabe unter gleichen Bedingungen 3.0 Österreich." - http://creativecommons.org/licenses/by-nc-sa/3.0/at/