

```

Sep 19 14:20:18 amd64 sshd[20494]: Accepted rsa for esser from ::ffff:87.234.201.207 port 61557
Sep 19 14:27:41 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 01:00:01 amd64 /usr/sbin/cron[29278]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 20 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 02:00:01 amd64 /usr/sbin/cron[31033]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 20 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:46:44 amd64 sshd[6516]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62004
Sep 20 12:46:44 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:48:41 amd64 sshd[6691]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62105
Sep 20 12:54:44 amd64 sshd[6694]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62514
Sep 20 15:27:35 amd64 sshd[9077]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64242
Sep 20 15:27:35 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:37:11 amd64 sshd[10102]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63375
Sep 20 16:37:11 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:38:10 amd64 sshd[10140]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63546
Sep 21 01:00:01 amd64 /usr/sbin/cron[17055]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 21 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 02:00:01 amd64 /usr/sbin/cron[17878]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 21 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:43:26 amd64 sshd[31088]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63397
Sep 21 17:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:53:39 amd64 sshd[31263]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64391
Sep 21 18:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 19:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 01:00:01 amd64 /usr/sbin/cron[24739]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 22 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 02:00:01 amd64 /usr/sbin/cron[25555]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 22 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 20:23:21 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 01:00:01 amd64 /usr/sbin/cron[24739]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 23 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 02:00:01 amd64 /usr/sbin/cron[25555]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 23 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:05 amd64 sshd[6554]: Accepted rsa for esser from ::ffff:193.168.1.5 port 59771 ssh2
Sep 23 18:04:05 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:34 amd64 sshd[6606]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62093
Sep 24 01:00:01 amd64 /usr/sbin/cron[12436]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 24 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 02:00:01 amd64 /usr/sbin/cron[13253]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 24 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 11:15:48 amd64 sshd[20989]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64456
Sep 24 11:15:48 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 13:49:08 amd64 sshd[21377]: Accepted rsa for esser from ::ffff:87.234.201.207 port 61330
Sep 24 13:49:08 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 15:42:07 amd64 kernel: snd_seq_midi_event: unsupported module, tainting kernel.
Sep 24 15:42:07 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 20:25:31 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 20:25:31 amd64 /usr/sbin/cron[29399]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62566
Sep 24 20:25:31 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 01:00:02 amd64 /usr/sbin/cron[16621]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 01:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 02:00:01 amd64 /usr/sbin/cron[14844]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 25 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:25 amd64 sshd[8889]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64183
Sep 25 10:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:47 amd64 sshd[8921]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64255
Sep 25 11:30:02 amd64 sshd[9372]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62029
Sep 25 11:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:05:37 amd64 sshd[11554]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62822
Sep 25 14:05:37 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:06:10 amd64 sshd[11586]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62951
Sep 25 14:07:17 amd64 sshd[11608]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63392
Sep 25 14:08:33 amd64 sshd[11630]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63709
Sep 25 15:25:33 amd64 sshd[12930]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62778

```

6. Inter-Prozess-Kommunikation (2)

6. IPC

6.2 Sockets

6.2.2 Exkurs: TCP/IP, UDP

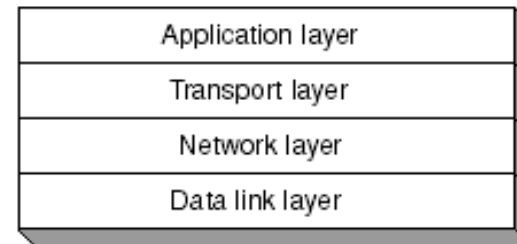
6.2.3 Sockets mit Python

6.3 Pipes

/home/esser/Daten/Dozent/Folien/bs-esser-18.doc

TCP/IP (1)

- Internet Protocol Suite
- Kombination verschiedener Protokolle
- vier Schichten (vgl. OSI-Schichten-Modell)



```

Sep 19 14:20:18 amd64 sshd[20494]: Accepted rsa for esser from ::ffff:87.234.201.207 port 61557
Sep 19 14:27:41 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 01:00:01 amd64 /usr/sbin/cron[29278]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 20 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 02:00:01 amd64 /usr/sbin/cron[31033]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 20 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:46:44 amd64 sshd[6516]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62004
Sep 20 12:46:44 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:48:41 amd64 sshd[6691]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62105
Sep 20 12:54:44 amd64 sshd[6694]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62514
Sep 20 15:27:35 amd64 sshd[9077]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64242
Sep 20 15:27:35 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:37:11 amd64 sshd[10102]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63375
Sep 20 16:37:11 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:38:10 amd64 sshd[10140]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63546
Sep 21 01:00:01 amd64 /usr/sbin/cron[17055]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 21 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 02:00:01 amd64 /usr/sbin/cron[17878]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 21 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:43:26 amd64 sshd[31088]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63397
Sep 21 17:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:53:39 amd64 sshd[31263]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64391
Sep 21 18:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 19:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 01:00:01 amd64 /usr/sbin/cron[24739]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 22 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 02:00:01 amd64 /usr/sbin/cron[25555]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 22 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 20:23:21 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 01:00:01 amd64 /usr/sbin/cron[24739]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 23 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 02:00:01 amd64 /usr/sbin/cron[25555]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 23 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:05 amd64 sshd[6554]: Accepted rsa for esser from ::ffff:193.168.1.5 port 59771 ssh2
Sep 23 18:04:05 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:34 amd64 sshd[6606]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62093
Sep 24 01:00:01 amd64 /usr/sbin/cron[12436]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 24 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 02:00:01 amd64 /usr/sbin/cron[13253]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 24 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 11:15:48 amd64 sshd[20989]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64456
Sep 24 11:15:48 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 13:49:08 amd64 sshd[21377]: Accepted rsa for esser from ::ffff:87.234.201.207 port 61330
Sep 24 13:49:08 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 15:42:07 amd64 kernel: snd_seq_midi_event: unsupported module, tainting kernel.
Sep 24 15:42:07 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 20:25:31 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 20:25:31 amd64 /usr/sbin/cron[29399]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62566
Sep 24 20:25:31 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 01:00:02 amd64 /usr/sbin/cron[16621]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 01:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 02:00:01 amd64 /usr/sbin/cron[14844]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 25 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:25 amd64 sshd[8889]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64183
Sep 25 10:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:47 amd64 sshd[8921]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64253
Sep 25 11:30:02 amd64 sshd[9372]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62029
Sep 25 11:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:05:37 amd64 sshd[11554]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62822
Sep 25 14:05:37 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:06:10 amd64 sshd[11586]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62951
Sep 25 14:07:17 amd64 sshd[11608]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63392
Sep 25 14:08:33 amd64 sshd[11630]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63709
Sep 25 15:25:33 amd64 sshd[12930]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62778

```

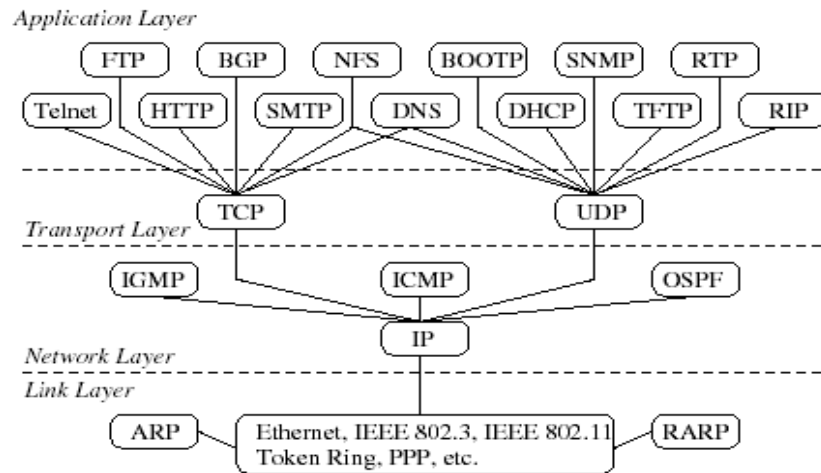
6.2 IPC: Kurzer Ausflug: TCP/IP, UDP

TCP/IP (2)

Die vier Layer

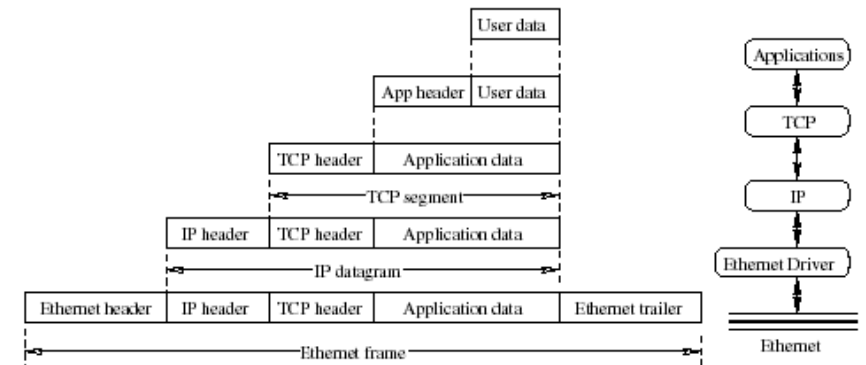
- **Data-Link-Schicht**
 - Service: Frames zuverlässig über eine Verbindung übertragen
 - Funktionen: Synchronisation, Fehlerkontrolle, Flusskontrolle
- **Netzwerkschicht**
 - Service: Beweg Pakete durch das Netzwerk
 - Funktionen: Routing, Adressierung, Switching, Überlast-Kontrolle
- **Transportschicht**
 - Service: Zustellung von Daten zwischen Rechnern überwachen
 - Funktionen: Verbindungsaufbau/-abbau, Fehlerkontrolle, Flusskontrolle
- **Applikationsschicht**
 - Service: behandelt Details des Anwendungsprogramms
 - Funktionen: alles applikationsspezifisch

TCP/IP (3)

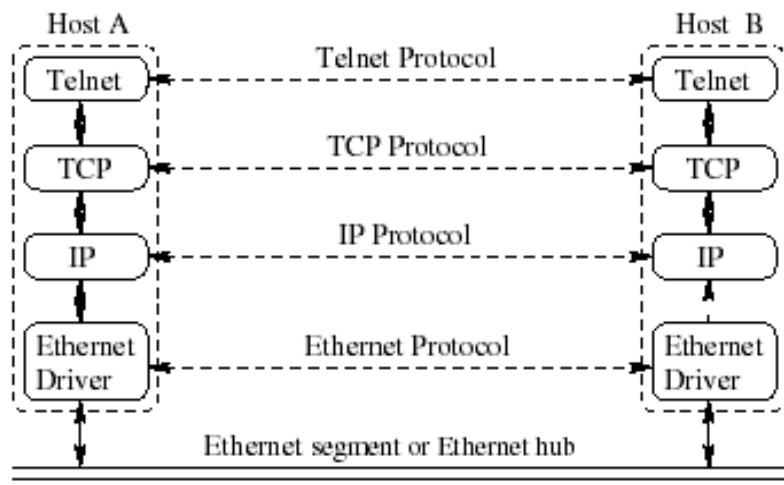


TCP/IP (5)

- Kapselung: auf jeder Ebene „einpacken“ (d.h. mit Protokoll-eigenem Header versehen) und auf die Reise schicken



TCP/IP (4)



TCP/IP (6)

Adressierung

- Server führen mehrere Dienste aus, Clients arbeiten mit verschiedenen Anwendungen, die gleichzeitig Anfragen stellen
 - Ordnung schaffen:
 - Port-Nummer
 - Rechnername
 - IP-Adresse
 - MAC-Adresse (Netzwerkkarte)
- Example: `imap.srv.fhm.edu:143`

TCP/IP (7)

Rechnernamen

- einen Rechner (im Internet oder im lokalen Netzwerk) eindeutig identifizieren
- benutzerfreundlich („sprechende“ Namen)
- hierarchisch organisiert
- Domain Name System (DNS): findet zu Rechnername die entsprechende IP-Adresse

TCP/IP (9)

Well-known port numbers: /etc/services (TCP & UDP)

#	0/tcp	Reserved	#	0/udp	Reserved
tcpmux	1/tcp	# TCP Port Service Multiplexer	tcpmux	1/udp	# TCP Port Service Multiplexer
compressnet	2/tcp	# Management Utility	compressnet	2/udp	# Management Utility
compressnet	3/tcp	# Compression Process	compressnet	3/udp	# Compression Process
#	4/tcp	# Unassigned	#	4/udp	# Unassigned
rje	5/tcp	# Remote Job Entry	rje	5/udp	# Remote Job Entry
#	6/tcp	# Unassigned	#	6/udp	# Unassigned
echo	7/tcp	Echo	echo	7/udp	Echo
#	8/tcp	# Unassigned	#	8/udp	# Unassigned
discard	9/tcp	# Discard	discard	9/udp	# Discard
#	10/tcp	# Unassigned	#	10/udp	# Unassigned
sysstat	11/tcp	users # Active Users	sysstat	11/udp	users # Active Users
#	12/tcp	# Unassigned	#	12/udp	# Unassigned
daytime	13/tcp	# Daytime (RFC 867)	daytime	13/udp	# Daytime (RFC 867)
#	14/tcp	# Unassigned	#	14/udp	# Unassigned
netstat	15/tcp	# Unassigned [was netstat]	#	15/udp	# Unassigned
#	16/tcp	# Unassigned	#	16/udp	# Unassigned
qotd	17/tcp	quote # Quote of the Day	qotd	17/udp	quote # Quote of the Day
msp	18/tcp	# Message Send Protocol	msp	18/udp	# Message Send Protocol
chargen	19/tcp	# Character Generator	chargen	19/udp	# Character Generator
ftp-data	20/tcp	# File Transfer [Default Data]	ftp-data	20/udp	# File Transfer [Default Data]
ftp	21/tcp	# File Transfer [Control]	ftp	21/udp	# File Transfer [Control]
ssh	22/tcp	# SSH Remote Login Protocol	ssh	22/udp	# SSH Remote Login Protocol
telnet	23/tcp	# Telnet	telnet	23/udp	# Telnet
#	24/tcp	any private mail system	#	24/udp	any private mail system
smtp	25/tcp	mail # Simple Mail Transfer	smtp	25/udp	mail # Simple Mail Transfer
#	26/tcp	# Unassigned	#	26/udp	# Unassigned
nsw-fe	27/tcp	# NSW User System FE	nsw-fe	27/udp	# NSW User System FE
#	28/tcp	# Unassigned	#	28/udp	# Unassigned
msg-icp	29/tcp	# MSG ICP	msg-icp	29/udp	# MSG ICP
#	30/tcp	# Unassigned	#	30/udp	# Unassigned

TCP/IP (8)

Port-Nummern

- Mehrere Dienste auf einem Rechner unterscheiden
- Mehrere parallele Verbindungen (vielleicht zu gleichem Dienst) zulassen
- **Port-Number**-Feld in TCP- oder UDP-Header
- „Well-known port numbers“: 0-1023
 - 1 - 255: Internet-weite Services
 - 256 - 1023: reserviert für Unix-spezifische Services
- Registrierte Ports: 1024-49151
- dynamische Ports: 49152-65535

TCP/IP (10)

IP-Adressen

- Jedes Rechner-Interface im Internet hat eine eindeutige IP-Adresse (manche Rechner auch mehrere)
- IPv4, 32 Bit, Notation mit Dezimalzahlen und Punkten

128.238.42.112 means

10000000 in 1st Byte
11101110 in 2nd Byte
00101010 in 3rd Byte
01110000 in 4th Byte

TCP/IP (11)

Internet Protocol (IP)

- Datagram (packet) protocol
- “Best-effort service”
 - Verlust
 - Vertauschen der Reihenfolge
 - Paket-Verdopplung
 - Verzögerung
- IP transportiert Daten **zwischen Maschinen, nicht zwischen Anwendungen**

TCP/IP (13)

IP-Service nicht ausreichend, darum: zusätzliche Dienste, die auf IP basieren

- **User Datagram Protocol (UDP)**
 - Prüfsummen
 - anwendungsspezifisch (mit Port-Nummern)
- **Transmission Control Protocol (TCP)**
 - zusätzlich zu UDP (Prüfsummen, Ports):
 - verlässlicher Bytestrom-Transport
 - Fluss- und Überlastungskontrolle

TCP/IP (12)

IP-Paket-Header:

0				1				2				3																											
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
Version				IHL				Type of Service				Total Length																											
Identification								Flags				Fragment Offset																											
Time to Live				Protocol				Header Checksum																															
Source Address																																							
Destination Address																																							
Options										Padding																													

TCP/IP (14)

UDP-Paket-Header

0				1				2				3																											
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
Source Port								Destination Port																															
Length								Checksum																															

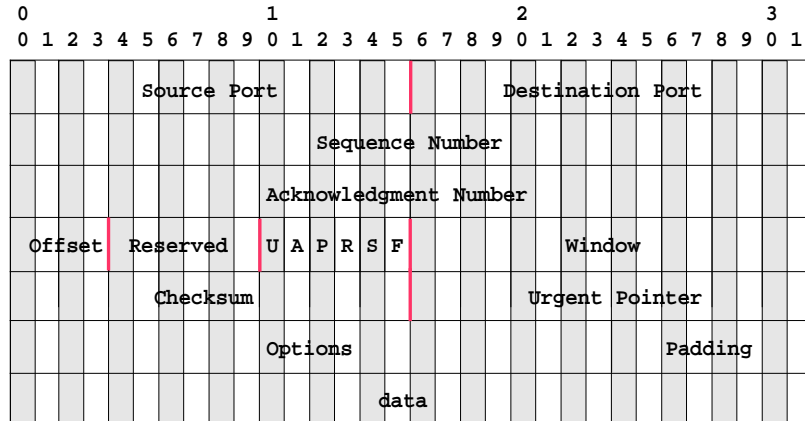
- Prüfsumme: wird aus UDP-Header, UDP-Daten und folgendem Pseudo-Header berechnet

32-bit Source IP Address															
32-bit Destination IP Address															
0x00				8-bit Protocol (0x17)				16-bit UDP Length							

- UDP-Paket ohne Adressangaben, weil es in IP-Paket verpackt wird

TCP/IP (15)

TCP-Paket-Header



TCP/IP (17)

- HTTP-Port nachschlagen:

```
$ grep http /etc/services | grep tcp
http                80/tcp             # World Wide Web HTTP
```

- also Port 80
- Sender (der Anfrage) wählt einen beliebigen freien Port, z. B. 50001
- Damit:
Absender = 192.168. 1. 2:50001
Empfänger = 129.187.244.212:80

TCP/IP (16)

- Versand von Daten über Sockets, definiert durch:
 - Absender (IP-Adresse und Port)
 - Empfänger (IP-Adresse und Port)
- Typisches Beispiel: Anfrage an einen Web-Server (HTTP) mit Server: www.fhm.edu (129.187.244.212)

TCP/IP (18)

3-Way-Handshake für Verbindungsaufbau

- Client sendet SYN
- Server antwortet SYN/ACK
- Client antwortet ACK
- danach steht die Verbindung

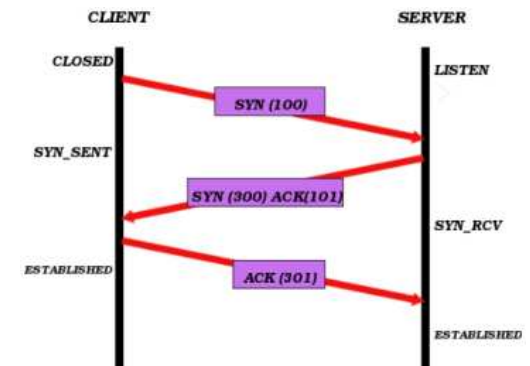


Bild: <http://www.cs.pub.ro/~rc/Laborator/lab5.html>

TCP/IP (19)

(1) 3-Way-Handshake

Frame 13 (74 bytes on wire, 74 bytes captured)
IP, Src Addr: 192.168.1.2 (192.168.1.2), Dst Addr: 129.187.244.212 (129.187.244.212)
TCP, Src Port: 10531 (10531), Dst Port: http (80), Seq: 0, Ack: 0, Len: 0
Source port: 10531 (10531)
Destination port: http (80)
Sequence number: 0 (relative sequence number)
Flags: 0x0002 (SYN)

Frame 14 (78 bytes on wire, 78 bytes captured)
IP, Src Addr: 129.187.244.212 (129.187.244.212), Dst Addr: 192.168.1.2 (192.168.1.2)
TCP, Src Port: http (80), Dst Port: 10531 (10531), Seq: 0, Ack: 1, Len: 0
Source port: http (80)
Destination port: 10531 (10531)
Sequence number: 0 (relative sequence number)
Acknowledgement number: 1 (relative ack number)
Flags: 0x0012 (SYN, ACK)

Frame 15 (66 bytes on wire, 66 bytes captured)
IP, Src Addr: 192.168.1.2 (192.168.1.2), Dst Addr: 129.187.244.212 (129.187.244.212)
TCP, Src Port: 10531 (10531), Dst Port: http (80), Seq: 1, Ack: 1, Len: 0
Source port: 10531 (10531)
Destination port: http (80)
Sequence number: 1 (relative sequence number)
Acknowledgement number: 1 (relative ack number)
Flags: 0x0010 (ACK)

(protokolliert mit Ethereal)

TCP/IP (20)

(2) HTTP-Anfrage

Frame 16 (542 bytes on wire, 542 bytes captured)
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:14:6c:99:3e:aa
IP, Src Addr: 192.168.1.2 (192.168.1.2), Dst Addr: 129.187.244.212 (129.187.244.212)
TCP, Src Port: 10531 (10531), Dst Port: http (80), Seq: 1, Ack: 1, Len: 476
Source port: 10531 (10531)
Destination port: http (80)
Sequence number: 1 (relative sequence number)
Next sequence number: 477 (relative sequence number)
Acknowledgement number: 1 (relative ack number)
Flags: 0x0018 (PSH, ACK)

Hypertext Transfer Protocol
GET /home/fhm/d_welcome.pcms HTTP/1.1\r\n
User-Agent: Opera/7.50 (X11; Linux i686; U) [en]\r\n
Host: www.fh-muenchen.de\r\n
Accept: text/html, application/xml;q=0.9, application/xhtml+xml, image/png, image/jpeg, image/gif, image/x-bitmap, */*;q=0.1\r\n
Accept-Language: de\r\n
Accept-Charset: iso-8859-1, utf-8, utf-16, */q=0.1\r\n
Accept-Encoding: deflate, gzip, x-gzip, identity, */q=0\r\n
Cache-Control: no-cache\r\n
Connection: Keep-Alive, TE\r\n
TE: deflate, gzip, chunked, identity, trailers\r\n
\r\n

(protokolliert mit Ethereal)

TCP/IP (21)

The screenshot shows the Ethereal interface with a packet list and packet details. The selected packet is a GET request for /home/fhm/d_welcome.pcms. The details pane shows the request headers, including User-Agent, Host, Accept, and Accept-Encoding. The packet bytes pane shows the raw data of the request.

(3) HTTP-Antwort

Frame 20 (1466 bytes on wire, 1466 bytes captured)
Ethernet II, Src: 00:14:6c:99:3e:aa, Dst: 00:00:00:00:00:00
IP, Src Addr: 129.187.244.212 (129.187.244.212), Dst Addr: 192.168.1.2 (192.168.1.2)
TCP, Src Port: http (80), Dst Port: 10531 (10531), Seq: 1, Ack: 477, Len: 1400
Source port: http (80)
Destination port: 10531 (10531)
Sequence number: 1 (relative sequence number)
Next sequence number: 1401 (relative sequence number)
Acknowledgement number: 477 (relative ack number)
Header length: 32 bytes
Flags: 0x0010 (ACK)

Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
Date: Sat, 16 Dec 2006 17:54:13 GMT\r\n
Server: Apache/1.3.3.1 (Unix) mod_perl/1.29 PHP/4.3.6 mod_ssl/2.8.17 OpenSSL/0.9.7d\r\n
X-Powered-By: PHP/4.3.6\r\n
Keep-Alive: timeout=15, max=100\r\n
Connection: Keep-Alive\r\n
Transfer-Encoding: chunked\r\n
Content-Type: text/html; charset=\r\n
\r\n
HTTP chunked response

(protokolliert mit Ethereal)

TCP/IP (23)

No.	Time	Source	Destination	Protocol	Info
13	1.603159	192.168.1.2	129.187.244.212	TCP	10531 > http [SYN] Seq=0 Ack=0 Win=5840 Len=0 MSS=1460 TSV=64198234
14	1.664595	129.187.244.212	192.168.1.2	TCP	http > 10531 [SYN, ACK] Seq=0 Ack=1 Win=25200 Len=0 TSV=1375405323
15	1.664658	192.168.1.2	129.187.244.212	TCP	10531 > http [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=641982402 TSER=13
16	1.664913	192.168.1.2	129.187.244.212	HTTP	GET /home/fhm/d_welcome.pcms HTTP/1.1
17	1.665012	192.168.1.2	129.187.244.212	TCP	10517 > http [FIN, ACK] Seq=0 Ack=0 Win=24214 Len=0 TSV=641982402 T
18	1.746846	129.187.244.212	192.168.1.2	TCP	http > 10531 [ACK] Seq=1 Ack=477 Win=25200 Len=0 TSV=1375405331 TSE
19	1.750239	129.187.244.212	192.168.1.2	TCP	http > 10517 [ACK] Seq=0 Ack=1 Win=25200 Len=0 TSV=1375405332 TSE=
20	1.947541	129.187.244.212	192.168.1.2	HTTP	HTTP/1.1 200 OK[Unreassembled Packet]
21	1.947597	192.168.1.2	129.187.244.212	TCP	10531 > http [ACK] Seq=477 Ack=1401 Win=8736 Len=0 TSV=641982685 TS
22	1.953441	129.187.244.212	192.168.1.2	HTTP	Continuation or non-HTTP traffic
23	1.953500	192.168.1.2	129.187.244.212	TCP	10531 > http [ACK] Seq=477 Ack=2001 Win=11632 Len=0 TSV=641982691 T
24	1.958799	129.187.244.212	192.168.1.2	HTTP	Continuation or non-HTTP traffic

Transmission Control Protocol, Src Port: http (80), Dst Port: 10531 (10531), Seq: 1, Ack: 477, Len: 1400

Hypertext Transfer Protocol

HTTP/1.1 200 OK\r\n

Date: Sat, 16 Dec 2006 17:54:13 GMT\r\n

Server: Apache/1.3.31 (Unix) mod_perl/1.29 PHP/4.3.6 mod_ssl/2.8.17 OpenSSL/0.9.7d\r\n

X-Powered-By: PHP/4.3.6\r\n

Keep-Alive: timeout=15, max=100\r\n

Connection: Keep-Alive\r\n

Transfer-Encoding: chunked\r\n

Content-Type: text/html; charset=\r\n

\r\n

HTTP chunked response

0130 20 05 60 75 0e 00 65 04 00 0a 43 61 0e 74 65 0e Chunked ..Conten

0140 74 2d 54 79 70 65 3a 20 74 65 78 74 2f 68 74 6d t-Type: text/htm

0150 6c 3b 20 63 68 61 72 73 65 74 3d 0d 0a 0d 0a 64 l: chars et....d

0160 36 62 0d 0a 3c 21 44 4f 43 54 59 50 45 20 48 54 6b.<IDO CTYPE HT

0170 4d 4c 20 50 55 42 4c 49 43 20 22 20 2f 2f 57 33 ML PUBLI C -//M3

0180 4b 2f 2f 44 54 44 20 48 54 4d 4c 20 34 2e 30 31 C//DIO H TML 4.01

0190 20 54 72 61 6e 73 69 74 69 6f 6e 61 6c 2f 2f 45 Transitioal//E

01a0 4e 22 3e 0d 0a 0d 0a 3c 48 54 4d 4c 3e 0d 0a 0d N">...< HTML...<

01b0 0a 3c 48 45 41 44 3e 0d 0a 3c 74 69 74 6c 65 3e <HEAD>.<title>

01c0 46 48 20 4d 26 75 75 6d 6c 3b 6e 63 68 65 6e 2c FH M&uum l;nchen.

01d0 20 48 6f 6d 65 70 61 67 65 3c 2f 74 69 74 6c 65 Homepag e</title

01e0 3e 0d 0a 3c 4d 45 54 41 20 48 54 50 2d 45 51 >.<META HTTP-Eq

01f0 55 49 56 3d 22 63 6f 6e 74 65 6e 74 2d 74 79 70 UIV=<con tent-type

0200 65 22 20 43 4f 4e 54 45 4e 54 3d 22 74 65 78 74 e" CONTE NT="text

0210 2f 68 74 6d 6c 3b 20 43 48 41 52 53 45 54 3d 69 /html; C HARSET=

0220 73 6f 2d 38 38 35 39 2d 31 22 3e 0d 0a 0d 0a 3c so-8859- 1">....<

Sockets in Python (2)

Socket erzeugen

Client

```
# INET / STREAM Socket erzeugen
s = socket.socket(
    socket.AF_INET,
    socket.SOCK_STREAM)
# Verbindung zu Port 80 aufbauen
# (der Standard-HTTP-Port)
s.connect(("www.fhm.edu", 80))
```

Server

```
# INET / STREAM Socket erzeugen
serversocket = socket.socket(
    socket.AF_INET,
    socket.SOCK_STREAM)
# Socket an Hostnamen und HTTP-
# Standard-Port 80 binden
serversocket.bind(
    (socket.gethostname(), 80) )
# zum Server-Socket werden;
# max. 5 parallele Verbindungen
serversocket.listen(5)
```

Quelle: <http://www.amk.ca/python/howto/sockets/>; angepasst

Sockets in Python (1)

Sockets benutzt man unter Python ähnlich wie in C:

C	Python
<code>sockfd = socket(AF_INET, SOCK_STREAM, 0);</code>	<code>sock = socket(AF_INET, SOCK_STREAM)</code>
<code>serv_addr.sin_addr.s_addr = INADDR_ANY;</code> <code>serv_addr.sin_family = AF_INET;</code> <code>serv_addr.sin_port = htons (port);</code>	
<code>bind(sockfd, (sockaddr *)&serv_addr,</code> <code>sizeof(serv_addr));</code>	<code>sock.bind ("", port)</code>
<code>listen(sockfd, queuelen);</code>	<code>sock.listen (queuelen)</code>
<code>con = accept(sockfd, ..., ...);</code>	<code>client, addr = sock.accept()</code>
<code>n = read(con, buf, sizeof(buf));</code> <code>write(con, buf, sizeof(buf));</code>	<code>client.recv()</code> <code>client.send()</code>

Sockets in Python (3)

Daten übertragen und Verbindung beenden

Client

```
# Anfrage an den Server schicken
request = "GET ..."
s.send (request)

# Antwort des Servers lesen
result = s.recv (MAXSIZE)

# Verbindung abbauen
s.shutdown (SHUT_RDWR)
s.close ()
```

Server

```
while 1:
    # Verbindungen annehmen
    (clientsocket, address) = serversocket.accept()
    # Anfrage in einem neuen Thread bearbeiten
    ct = client_thread(clientsocket)

def client_thread (sock):
    start_new_thread (client_handler, sock)

def client_handler (sock):
    request = sock.recv ()
    ... # Anfrage bearbeiten
    sock.send (antwort)
    sock.shutdown (SHUT_RDWR); sock.close ()
```

Dokumentation: <http://docs.python.org/lib/module-socket.html>

```

Sep 19 14:27:41 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 01:00:01 amd64 /usr/sbin/cron[29278]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 20 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 02:00:01 amd64 /usr/sbin/cron[30103]: (root) CMD (/sbin/evlogmgr -c "age > *30d")
Sep 20 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:46:44 amd64 sshd[6561]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62004
Sep 20 12:46:44 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:48:41 amd64 sshd[6609]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62105
Sep 20 12:54:44 amd64 sshd[6604]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62044
Sep 20 15:27:35 amd64 sshd[9077]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64242
Sep 20 15:27:35 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:37:11 amd64 sshd[10102]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63375
Sep 20 16:37:11 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:38:10 amd64 sshd[10140]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63546
Sep 21 01:00:01 amd64 /usr/sbin/cron[17051]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 21 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 02:00:01 amd64 /usr/sbin/cron[17878]: (root) CMD (/sbin/evlogmgr -c "age > *30d")
Sep 21 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:43:26 amd64 sshd[31898]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63397
Sep 21 17:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:53:39 amd64 sshd[31269]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64391
Sep 21 18:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 19:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 01:00:01 amd64 /usr/sbin/cron[4674]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 22 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 02:00:01 amd64 /usr/sbin/cron[5499]: (root) CMD (/sbin/evlogmgr -c "age > *30d")
Sep 22 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 20:23:21 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 01:00:01 amd64 /usr/sbin/cron[24739]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 23 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 02:00:01 amd64 /usr/sbin/cron[25555]: (root) CMD (/sbin/evlogmgr -c "age > *30d")
Sep 23 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:05 amd64 sshd[8554]: Accepted publickey for esser from ::ffff:192.168.1.5 port 59771 [n2]
Sep 23 18:04:05 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:34 amd64 sshd[6606]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62093
Sep 24 01:00:01 amd64 /usr/sbin/cron[12436]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 24 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 02:00:01 amd64 /usr/sbin/cron[13231]: (root) CMD (/sbin/evlogmgr -c "age > *30d")
Sep 24 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 11:15:48 amd64 sshd[20998]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64456
Sep 24 11:15:48 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 13:49:08 amd64 sshd[23197]: Accepted rsa for esser from ::ffff:87.234.201.207 port 61330
Sep 24 13:49:08 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 15:42:07 amd64 kernel: amd_seg_oss: unsupported module, tainting kernel.
Sep 24 15:42:07 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 15:42:07 amd64 kernel: amd_seg_oss: unsupported module, tainting kernel.
Sep 24 20:25:11 amd64 sshd[2939]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62566
Sep 24 20:25:11 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 01:00:02 amd64 /usr/sbin/cron[662]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 01:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 02:00:01 amd64 /usr/sbin/cron[1484]: (root) CMD (/sbin/evlogmgr -c "age > *30d")
Sep 25 02:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:25 amd64 sshd[8889]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64183
Sep 25 10:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:47 amd64 sshd[8921]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64253
Sep 25 11:30:02 amd64 sshd[9372]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62029
Sep 25 11:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:05:37 amd64 sshd[11554]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62822
Sep 25 14:05:37 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:06:10 amd64 sshd[11586]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62951
Sep 25 14:07:17 amd64 sshd[11608]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63392
Sep 25 14:08:33 amd64 sshd[11630]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63709
Sep 25 15:25:33 amd64 sshd[12930]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62778

```

6.3 IPC: Pipes

Pipes (1)

- Zum Kennenlernen: Pipes in der Shell
- Standardausgabe eines Programms zur Standardeingabe des nächsten Programms machen
- Verkettung mehrerer (Filter-) Anwendungen

Ohne Pipe – mit temp. Dateien

```

find ./ -name "*.txt" >/tmp/t1
grep -i "betriebssystem" </tmp/t1 >/tmp/t2
sort </tmp/t2
rm /tmp/t1 /tmp/t2

```

Mit Pipe

```

find ./ -name "*.txt" | grep -i \
"betriebssystem" | sort

```

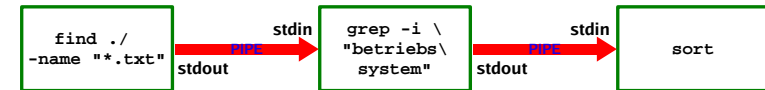
Pipes (2)

- Analyse:

```

find ./ -name "*.txt" | grep -i "betriebssystem" | sort

```



- Die Pipe ist ein **unidirektionaler** Kommunikationsmechanismus
- Es gibt benannte und unbenannte Pipes

Unbenannte Pipes (1)

- Vorgehensweise bei unbenannten Pipes immer gleich:
 - Prozess erzeugt Pipe (Anfangs- und Endstück)
 - Prozess verdoppelt sich mit **fork()**
 - Vater und Sohn verwenden die beiden Enden der Pipe zur (Ein-Weg-) Kommunikation
- Pipe bleibt erhalten (offen), bis einer der Prozesse sie explizit mit **close()** schließt
- Prozesse, die keine „Verwandschaft“ (fork) haben, brauchen benannte Pipes

Unbenannte Pipes (2)

- In Pipe schreiben und lesen wie in Datei

```
int pipe_fds[2]; // File-Deskriptoren für die beiden
                // Enden der Pipe

pipe(pipe_fds); // Erzeugen einer Pipe;
               // pipe_fds[0]: zum Lesen
               // pipe_fds[1]: zum Schreiben

if (fork() != 0) { // Vaterprozess:
    // Daten erzeugen und durch die Pipe schicken
    write(pipe_fds[1], data, ...);
} else { // Kindprozess:
    // Daten aus der Pipe lesen
    read(pipe_fds[0], data, ...);
}
```

Benannte Pipes (2)

- anschließend lesen und schreiben die beiden Prozesse ebenfalls wie bei Dateien:

Server (liest)

```
#define PIPENAME "/tmp/mypipe"
int main() {
    int fd, ret_val, count, numread;
    char buf[MAX_BUF_SIZE];

    /* benannte Pipe erzeugen */
    ret_val = mkfifo(PIPENAME, 0666);

    if ((ret_val == -1) && (errno != EEXIST)) {
        perror("Fehler beim Erzeugen der Pipe");
        exit(1);
    }

    /* Pipe zum LESEN öffnen */
    fd = open(PIPENAME, O_RDONLY);

    /* aus der Pipe lesen */
    numread = read(fd, buf, MAX_BUF_SIZE);

    printf("Half Duplex Server: Aus Pipe gelesen: %sn", buf);
}
```

Client (schreibt)

```
#define PIPENAME "/tmp/mypipe"
int main(int argc, char *argv[]) {
    int fd;

    /* Aufrufparameter auslesen */
    if (argc != 2) {
        printf("Usage : %s <string>\n",
            argv[0]);
        exit(1);
    }

    /* Pipe zum SCHREIBEN öffnen */
    fd = open(PIPENAME, O_WRONLY);

    /* In die Pipe schreiben */
    write(fd, argv[1], strlen(argv[1]));
}
```

Benannte Pipes (1)

- Mit benannten Pipes ist auch Kommunikation zwischen „nicht verwandten“ Prozessen möglich:
- Prozesse legen gemeinsamen Namen fest (vgl. benannte Semaphore etc.)
`int mkfifo(const char *pathname, mode_t mode);`
das erzeugt eine neue FIFO-Spezialdatei im Dateisystem
- Prozesse öffnen Pipe wie eine normale Datei

Pipes (7)

- Kommunikation in beide Richtungen: zwei Pipes verwenden

```
#define PIPE_A "/tmp/mypipe-a"
#define PIPE_B "/tmp/mypipe-b"

int main() {
    int fd, ret_val, count, numread;
    char buf[MAX_BUF_SIZE];

    /* benannte Pipes erzeugen */
    ret_val = mkfifo(PIPE_A, 0666);
    if ((ret_val == -1) && (errno != EEXIST)) { ... }
    ret_val = mkfifo(PIPE_B, 0666);
    if ((ret_val == -1) && (errno != EEXIST)) { ... }

    /* Pipes zum LESEN bzw. SCHREIBEN öffnen */
    fd_a = open(PIPE_A, O_RDONLY);
    fd_b = open(PIPE_B, O_WRONLY);

    /* mit Pipes arbeiten */
    numread = read(fd_a, buf, MAX_BUF_SIZE);
    write(fd_b, buf, ...);
}
```

Windows-Pipes (1)

Benannte Pipes: zwei Möglichkeiten

- Für Unix-Kompatibilität:

```
_pipe()
```

mit gleicher Syntax wie Unix `pipe()`
(`open`, `read`, `write`)

- Standard-Pipes:

```
#define FIFO_NAME "\\.\pipe\mypipe"  
hPipe = CreateNamedPipe(FIFO_NAME, ...)
```

Windows-übliche Objekte

Windows-Pipes (3)

```
fConnected = ConnectNamedPipe(hPipe, NULL) ?  
    TRUE : (GetLastError() == ERROR_PIPE_CONNECTED);  
  
printf("Process %d opening FIFO_WRONLY\n", getpid());  
  
if (fConnected) {  
    while(bytes_sent < TEN_MEG) {  
        if (!WriteFile(  
            hPipe,           // handle to file  
            buffer,         // data buffer  
            BUFFER_SIZE,    // number of bytes to write  
            &NumberOfBytesWritten, // number of bytes written  
            NULL)           // overlapped buffer  
        ) {  
            fprintf(stderr, "Write error on pipe\n");  
            exit(EXIT_FAILURE);  
        }  
  
        bytes_sent += NumberOfBytesWritten;  
    }  
  
    printf("Process %d finished\n", getpid());  
} else {  
    // The client could not connect, so close the pipe.  
    CloseHandle(hPipe);  
}
```

```
hPipe1 = CreateFile(  
    FIFO_NAME,           // Open the FIFO  
    GENERIC_READ,       // open for reading  
    0,                  // share for writing  
    NULL,               // no security  
    OPEN_EXISTING,     // existing file only  
    FILE_ATTRIBUTE_NORMAL, // normal file  
    NULL);             // no attr. template  
if (hPipe1 == INVALID_HANDLE_VALUE) {  
    printf("Could not create file %s\n", FIFO_NAME);  
    exit(EXIT_FAILURE);  
}  
printf("Process %d result %X\n", getpid(), hPipe1);  
do {  
    ReadFile(  
        hPipe1,         // handle to file  
        buffer,        // data buffer  
        BUFFER_SIZE,   // number of bytes to read  
        &NumberOfBytesRead, // number of bytes read  
        NULL           // overlapped buffer  
    );  
    bytes_read += NumberOfBytesRead;  
} while (NumberOfBytesRead > 0);  
CloseHandle(hPipe1);
```

Windows-Pipes (2)

```
#include <stdio.h>  
#include <stdlib.h>  
#include <windows.h>  
#include <io.h>  
#include <fcntl.h>  
#include <process.h>  
#include <string.h>  
#define FIFO_NAME  
    "\\.\pipe\mypipe"  
#define BUFSIZE 1024  
#define PIPE_TIMEOUT 5000 // 5 sec.  
#define F_OK 0  
  
void main() {  
    int fd;  
    HANDLE hPipe;  
    // Check if the FIFO exists and create it if necessary.  
    if (_access(FIFO_NAME, F_OK) == -1) {  
        hPipe = CreateNamedPipe(  
            FIFO_NAME,           // pipe name  
            PIPE_ACCESS_DUPLEX, // read/write access  
            PIPE_TYPE_MESSAGE | // message type pipe  
            PIPE_READMODE_MESSAGE | // message-read mode  
            PIPE_WAIT,          // blocking mode  
            PIPE_UNLIMITED_INSTANCES, // max. instances  
            BUFSIZE,           // output buffer size  
            BUFSIZE,          // input buffer size  
            PIPE_TIMEOUT,      // client time-out  
            NULL);            // no security attribute  
        if (hPipe == INVALID_HANDLE_VALUE) {  
            fprintf(stderr, "Could not create fifo %s\n", FIFO_NAME);  
            exit(EXIT_FAILURE);  
        }  
    }  
    printf("Process %d opening FIFO\n", getpid());  
    // Open FIFO and output status result  
    fd = open(FIFO_NAME, O_RDONLY);  
    printf("Process %d file descriptor: %d\n", getpid(), fd);  
    Sleep(PIPE_TIMEOUT);  
    // Close FIFO  
    if (fd != -1)  
        (void)close(fd);  
    printf("Process %d finished\n", getpid());  
    exit(EXIT_SUCCESS);  
}
```

Quelle: http://msdn2.microsoft.com/en-gb/library/ms811896.aspx#ucmgch09_topic13h