

```

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]: SFATS: 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]: SFATS: 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]: SFATS: 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]: SFATS: dropped 0
Sep 20 12:48:41 amd64 sshd[6099]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62105
Sep 20 12:54:44 amd64 sshd[6094]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62514
Sep 20 13:27:25 amd64 sshd[9077]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64242
Sep 20 15:27:35 amd64 syslog-ng[7653]: SFATS: 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]: SFATS: dropped 0
Sep 20 16:38:10 amd64 sshd[9140]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63546
Sep 21 01:00:01 amd64 /usr/sbin/cron[34545]: (root) CMD (/sbin/evlogmgr -c 'severity=DEBUG')
Sep 21 01:00:01 amd64 /usr/sbin/cron[7653]: SFATS: dropped 0
Sep 21 02:00:01 amd64 /usr/sbin/cron[17308]: (root) CMD (/sbin/evlogmgr -c 'age > "30d"')
Sep 21 02:00:01 amd64 /usr/sbin/cron[17308]: (root) CMD (/sbin/evlogmgr -c 'age > "30d"')
Sep 21 17:43:26 amd64 sshd[31263]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64391
Sep 21 17:43:26 amd64 syslog-ng[7653]: SFATS: dropped 0
Sep 21 17:43:26 amd64 sshd[31263]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64391
Sep 21 18:43:26 amd64 syslog-ng[7653]: SFATS: dropped 0
Sep 21 18:43:26 amd64 sshd[31263]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64391
Sep 21 19:43:26 amd64 syslog-ng[7653]: SFATS: dropped 0
Sep 22 01:00:01 amd64 /usr/sbin/cron[46744]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 22 01:00:01 amd64 syslog-ng[7653]: SFATS: dropped 0
Sep 22 02:00:01 amd64 /usr/sbin/cron[1945]: (root) CMD (/sbin/evlogmgr -c 'age > "30d"')
Sep 22 02:00:01 amd64 syslog-ng[7653]: SFATS: dropped 0
Sep 22 20:23:21 amd64 syslog-ng[7653]: SFATS: dropped 0
Sep 23 01:00:01 amd64 /usr/sbin/cron[2473]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 23 01:00:01 amd64 syslog-ng[7653]: SFATS: dropped 0
Sep 23 02:00:01 amd64 /usr/sbin/cron[2535]: (root) CMD (/sbin/evlogmgr -c 'age > "30d"')
Sep 23 02:00:01 amd64 syslog-ng[7653]: SFATS: dropped 0
Sep 23 18:04:05 amd64 sshd[6554]: Accepted publickey for esser from ::ffff:192.168.1.5 port 59771 ssh2
Sep 23 18:04:05 amd64 syslog-ng[7653]: SFATS: dropped 0
Sep 23 18:04:34 amd64 sshd[6066]: 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]: SFATS: dropped 0
Sep 24 02:00:01 amd64 /usr/sbin/cron[13531]: (root) CMD (/sbin/evlogmgr -c 'age > "30d"')
Sep 24 02:00:01 amd64 syslog-ng[7653]: SFATS: 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]: SFATS: 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]: SFATS: 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]: SFATS: dropped 0
Sep 24 20:25:31 amd64 sshd[29399]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62566
Sep 24 20:25:31 amd64 syslog-ng[7653]: SFATS: dropped 0
Sep 25 01:00:02 amd64 /usr/sbin/cron[621]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 01:00:02 amd64 syslog-ng[7653]: SFATS: dropped 0
Sep 25 02:00:02 amd64 /usr/sbin/cron[14841]: (root) CMD (/sbin/evlogmgr -c 'age > "30d"')
Sep 25 02:00:02 amd64 syslog-ng[7653]: SFATS: 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]: SFATS: 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]: SFATS: 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]: SFATS: 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[11606]: 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

```

9 Projekt: Web-Proxy

Projekt: Grundlagen

Übersicht

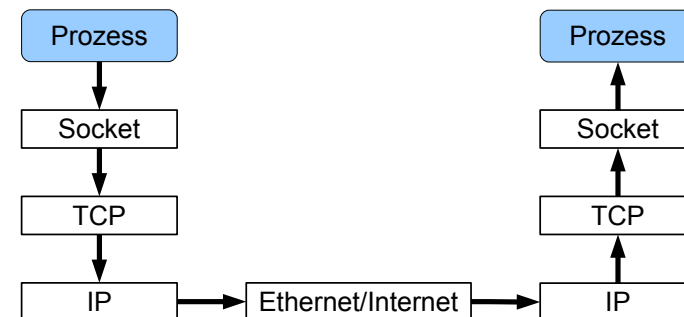
- Datenstrukturen für Sockets
- socket(), bind(), listen(), accept()
- htons(), ntohs(), inet_ntoa(), inet_aton()
- Socket-Deskriptoren als File-Deskriptoren
- Basics zu HTTP

Programmier-Projekt

TCP, IP & Co. (1)

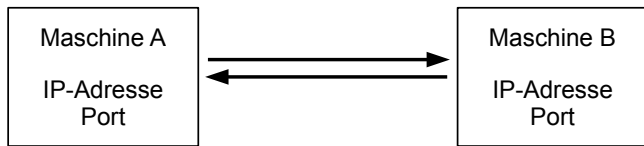
- Implementierung eines Web-Proxys mit folgenden Features
 - HTTP Proxy (GET, HEAD, POST)
 - Keep-Alive (persistente Verbindung)
 - Caching im RAM plus HEAD-Anfrage
 - Frei wählbare Ports für Proxy-Dienst und Steuerung / Konfiguration
 - beliebig viele parallele Verbindungen, über Threads realisiert
 - Server beendet sich/gibt Statusinfo, wenn das Dokument /exit bzw. /status angefordert wird
 - ggf. weitere Funktionen

- Hier keine Theorie zu Netzwerken, Layers etc.
- Sockets erlauben Kommunikation über TCP/IP



TCP, IP & Co. (2)

- TCP (Transmission Control Protocol): verbindungsorientiert, zuverlässig; nutzt IP (Internet Protocol)
- TCP-Verbindung: (IP₁, Port₁, IP₂, Port₂)



(Port nötig, da mehrere Verbindungen möglich)

TCP, IP & Co. (4)

- `netstat -a` zeigt zusätzlich Server an, d. h., Sockets, die auf Verbindungsanfragen warten:

```
[esser@macbookpro:C-Projekte]$ netstat -a
Active Internet connections (servers and established)
Proto Rec-Q Snd-Q Local Address Foreign Address State
tcp4 0 0 macbookpro.59433 fauils219.inform.https ESTABLISHED
tcp4 0 0 *.http-alt *.* LISTEN
tcp4 0 0 macbookpro.59432 channelproxy-shv.https ESTABLISHED
[...]
```

TCP, IP & Co. (3)

- `netstat` zeigt (u. a.) offene TCP-Verbindungen an, bekannte Ports erscheinen mit Namen

```
[esser@macbookpro:C-Projekte]$ netstat
Active Internet connections (w/o servers)
Proto Rec-Q Snd-Q Local Address Foreign Address State
tcp4 0 0 macbookpro.59433 fauils219.inform.https ESTABLISHED
tcp4 0 0 macbookpro.59432 channelproxy-shv.https ESTABLISHED
tcp4 0 0 macbookpro.59428 edge-star-shv-07.https ESTABLISHED
tcp4 0 0 macbookpro.59348 my.ohmportal.de.imaps ESTABLISHED
tcp4 0 0 macbookpro.59347 imap.1und1.de.imap2 ESTABLISHED
tcp4 0 0 macbookpro.59241 173.194.116.149.https ESTABLISHED
[...]
```

```
[esser@macbookpro:~]$ egrep "^imap|^https" /etc/services | grep tcp
imap2 143/tcp # Internet Message Access Protocol
https 443/tcp # http protocol over TLS/SSL
imaps 993/tcp # imap4 protocol over TLS/SSL
```

Datenstrukturen

- Socket-Deskriptoren
 - einfache Integers, wie File-Deskriptoren
 - `socket()` und `accept()` geben Socket-Deskriptoren zurück

- IP-Adressen

```
struct sockaddr_in {
    short sin_family; // Typ, z. B. AF_INET
    unsigned short sin_port; // Port, z. B. htons (8080)
    struct in_addr sin_addr; // kodierte Adresse
    char sin_zero[8]; // freier Platz
};

struct in_addr {
    unsigned int s_addr; // mit inet_aton() füllen
                        // oder auf INADDR_ANY setzen
                        // 32 Bit für IP-Adresse
};
```

Sockets erzeugen

- Es gibt zwei Arten von Sockets (für unsere Zwecke)
 - generischer Socket, der mit `socket ()` erzeugt wird → kann verwendet werden,
 - um einen TCP-Port zu binden; bleibt dann dauerhaft „in Betrieb“ (Server)
 - um eine Verbindung zu einem Server aufzubauen (Client)
 - Verbindungs-Socket, der mit `accept ()` erzeugt wird → ist nur für eine konkrete Verbindung (mit einem Client) zuständig

TCP-Server erzeugen (1)

```
int sd; // socket descriptor for server
struct sockaddr_in server;

sd = socket (PF_INET, SOCK_STREAM, 0);
// PF_INET: Protocol Family, Internet
// SOCK_STREAM: TCP

server.sin_port = htons (PORT);
server.sin_addr.s_addr = INADDR_ANY;
server.sin_family = AF_INET;
// Address Family, Internet
```

Sockets erzeugen

- Server
 - `sd = socket ()`
 - Server-Adresse konfigurieren
 - `bind (sd, Server-Adresse)`
 - `listen (sd)`
 - `conn = accept (sd, &Client-Adresse)`
(zweiter Socket!)
- Client
 - `sd = socket ()`
 - Ziel-Adresse konfigurieren
 - `connect (sd, &Ziel-Adresse)`

TCP-Server erzeugen (2)

```
int conn; // socket descriptor for connection
#define SOCKADDR_SIZE sizeof(struct sockaddr_in)
int clilen = SOCKADDR_SIZE;
struct sockaddr_in client;

bind (sd, (struct sockaddr *)&server,
      SOCKADDR_SIZE)

listen (sd, 0);

conn = accept (sd, (struct sockaddr *) &client,
              &clilen);
```

TCP-Server erzeugen (3)

```
char buf[bufsize];

// lesen: wie aus Datei
readbytes = read (conn, &buf, bufsize);

// schreiben: wie in Datei
write (conn, &buf, n);

// diese Verbindung schliessen
shutdown (conn, SHUT_RDWR);
close (conn);

// auch dup2() funktioniert mit socket descr.
```

send(), recv()

- Statt write und read besser send() und recv() verwenden → speziell für Sockets gedacht
- write (sd, buffer, length);
→
send (sd, buffer, length, flags);
- read (sd, buffer, length);
→
recv (sd, buffer, length, flags);
- jeweils mit flags = 0 (siehe Manpages)

TCP-Client erzeugen

```
int sd; // socket descriptor for server
struct sockaddr_in client;
sd = socket (PF_INET, SOCK_STREAM, 0);

client.sin_port = htons(PORT);
client.sin_addr.s_addr = inet_addr("192.1.2.3");
client.sin_family = AF_INET;
// Address Family, Internet

int res = connect (sd, (struct sockaddr *)&client,
                  sizeof(client));
write (sd, request, strlen(request));
int n = read (sd, response, sizeof(response));
```

htons(), ntohs()

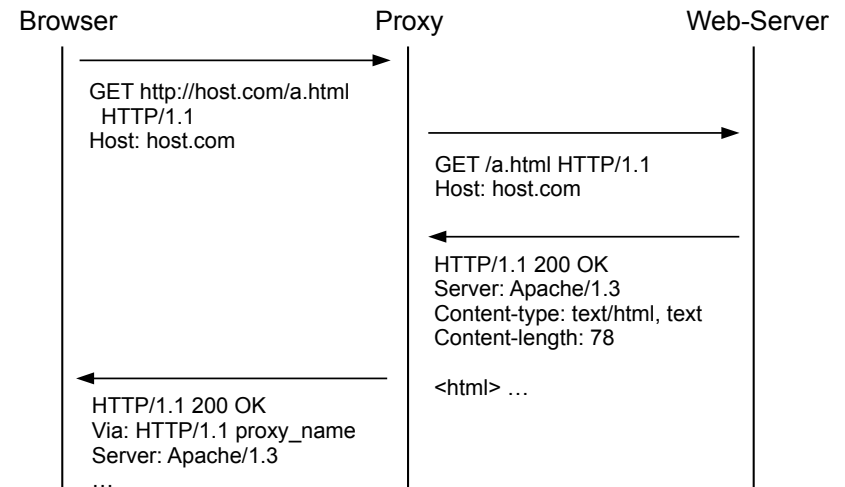
- TCP/IP gibt Standard-Byte-Order vor (Network Byte Order, Big-Endian, most-significant byte first)
- Linux-PC verwendet Little-Endian (least-significant byte first)
- htons() und ntohs() konvertieren Portnummern: **host to network** bzw. **network to host**
- darum:
server.sin_port = **htons**(PORT);

inet_ntoa()

- Aus Adress-Eintrag IP-Adresse auslesen mit `inet_ntoa()`:

```
struct sockaddr_in client;  
accept (sd, (struct sockaddr *) &client,  
        &clilen);  
printf ("Client-Adresse: %s \n",  
        inet_ntoa (client.sin_addr));
```
- wertet `client.sin_addr.s_addr` aus

Proxy-Grundlagen



HTTP Basics

- TCP-Verbindung über Sockets, ASCII-Protokoll
- Client → Server:

```
GET /index.html HTTP/1.1  
Host: domainname.top  
User-Agent: Mozilla/5.0 (...)
```

- Server → Client:

<pre>HTTP/1.1 201 OK Content-Type: text/html <html> ...</pre>	<pre>HTTP/1.1 404 Not found Content-Type: text/html <html> ...</pre>
--	---

Proxy-Grundlagen

- Browser schickt an Proxy-Server eine Anfrage der folgenden Form:

```
GET http://google.de/ HTTP/1.1  
Host: google.de  
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:40.0  
.0) Gecko/20100101 Firefox/40.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,  
*/*;q=0.8  
Accept-Language: de,en-US;q=0.7,en;q=0.3  
Accept-Encoding: gzip, deflate  
Cookie: SID=DQAAAF [...]  
Connection: keep-alive
```

- Anders als bei normaler HTTP-Anfrage:
enthält Domain (`http://google.de/` statt `/`)