IPV4 und IPv6 Grundlagen

IPv6 (Internet Protocol Version 6)

- 1. Ethernet und IP
- 2. Protokolle
- 3. Sicherheit
- 4. Programmierung
- 5. Feedback

1. Ethernet und IP

- 1.1. Ethernet
- 1.2. IPv4 Adressnotation
- 1.3. IPv6 Adressnotation
- 1.4. IPv4 Address Resolution Protocol (ARP)
- 1.5. IPv6 Neighbor Discovery Protocol (NDP)
- 1.6. Dual-Stack
- 1.7. Gegenüberstellung

1.1. Ethernet

- Mac-Adresse

  Im Netzwerk werden alle Pakete mit dem Ethernet-Protokoll übertragen.

  Hier wird nur die sog. Mac-Adresse verwendet. Sie ist 6 Bytes lang.

  Jeder Netzwerkkarte hat ihre eigne individuelle "Mac-Adresse".

  In einem Netzwerk darf die Mac-Adresse nie zweimal vorkommen.

1.1.1. Unicast / Broadcast

- Unicast

  Normalerweise sendet eine Karte immer direkt zu der Adresse einer anderen Karte.

- Broadcast

  Ein Paket mit der Mac-Adresse ff:ff:ff:ff:ff:ff

  Dieses Paket wird von allen Netzwerkkarten empfangen.

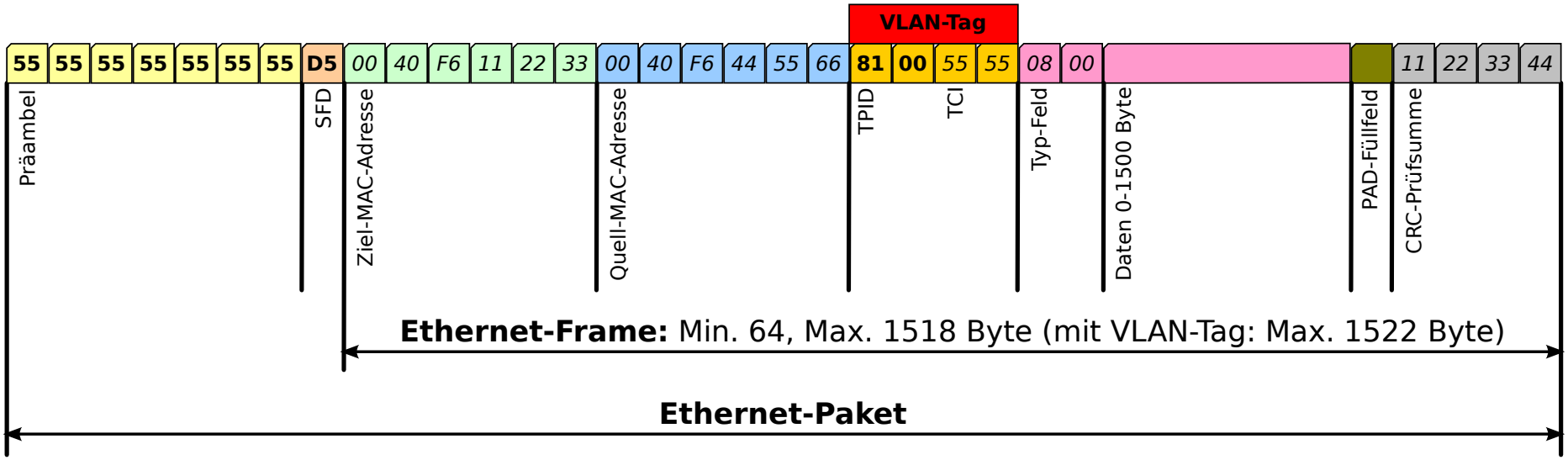
1.1.2. Multicast

- Multicast

  Pakete mit z.B. den Mac-Adressen 01:00:5e:**:**:** oder 33:33:**:**:**:**

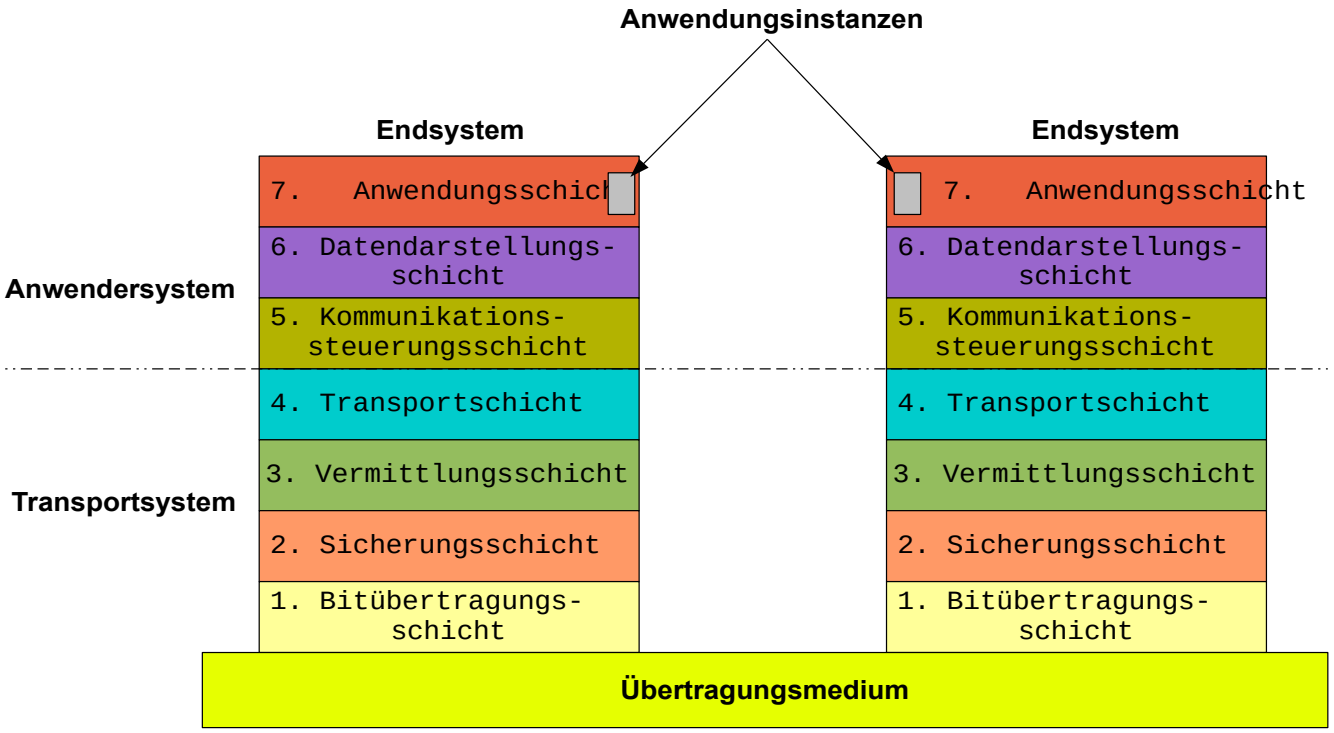
  Diese Pakete werden von einer beliebigen Anzahl Netzwerkkarten empfangen.

1.1.3. Aufbau eines Ethernet Paketes



Quelle: http://de.wikipedia.org/wiki/Ethernet

1.1.4. Schichtenmodell



Quelle: http://de.wikipedia.org/wiki/OSI-Modell

1.1.4. Schichtenmodell IP

Schicht 4 Transportschicht	UCP, TCP, SCTP
Schicht 3 Vermittlungsschicht	ICMP, ICMPv6
Schicht 3 Vermittlungsschicht	IPv4, IPV6
	ARP
Schicht 2 Sicherungsschicht	Ethernet
Schicht 1 Bitübertragungsschicht	Ethernet

1.1.4.1. Bridge und Router, Hub und Switch

- Das Ethernet kann über Hubs und Switche verlängert werden.
- Das Ethernet endet an einem Router.
- Eine Bridge kann das Ethernet erweitern.

1.1.4.2. Routing

- Rechner im gleichen IP-Netz werden direkt im Ethernet gesucht.
- Wenn eine Route eingetragen ist, werden die Pakete an das Gateway weitergeleitet.
- Das Gateway muss direkt im IP-Netz erreichbar sein.
- Pakete für alle anderen Rechner werden an das Gateway gesendet.

1.1.5. VLAN (Virtual Local Area Network)

- VLAN muss auf den Switches konfiguriert werden.
- Trennung des Ethernet in eigenen Bereiche.
- Man kann so sichere und unsichere Netze trennen.

1.1.5. VLAN (Virtual Local Area Network)

- Beispiel:

  Die Netzwerkdose in einem Konferenzraum wird mit VLAN am Switch eingerichtet. In diesem VLAN ist nur ein Router für die Verbindung ins Internet erreichbar, nicht jedoch das Intranet oder die Dateifreigaben der Server.

1.1.5.1. VLAN Access Port

- Ein Access Port hat nur ein einziges VLAN.
- Die Pakete kommen ohne ein VLAN-Tag rein.
- Der Switch fügt intern in das Paket das VLAN-Tag ein.
- Ausgehend wird das VLAN-Tag von den Pakten entfernt.
- Die Gegenstelle am Switch-Port sieht nur ein normales Ethernet.

1.1.5.2. VLAN Trunk Port

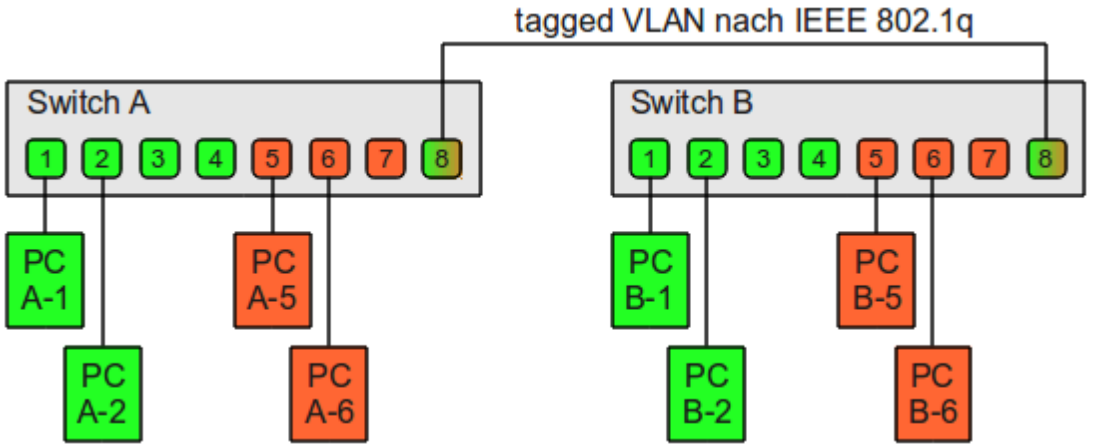
- Ein Trunk Port hat viele VLANs.
- Die Pakete kommen immer mit VLAN-Tag auf dem Port rein.
- Ausgehend wird immer ein VLAN-Tag mit übertragen.
- Die Gegenstelle am Switch-Port muss die gleiche VLAN Konfiguration haben.

1.1.5.3. VLAN Hybrid Port

- Ein Hybrid Port ist ein erweiterter Trunk Port
- Zusätzlich werden auch Pakete ohne VLAN-Tag verarbeitet.
- Die Gegenstelle am Switch-Port muss die gleiche VLAN Konfiguration haben.

1.1.5.4. VLAN Beispiel

- Hier ein Beispiel mit Access Ports und Trunk Ports.



Quelle: Werner Fischer, Thomas Krenn Wiki

1.1.5.5. Stacked VLAN, Q in Q

- Ein Paket mit Stacked-VLAN hat 2 VLAN-Tags
- Das "Outer" oder "Service" VLAN bestimmt den Transport
- Das "Inner" oder "Customer" VLAN wird mitgeführt
- Damit kann man VLAN-Netze ineinander verschachteln.

1.1.5.5. VLAN Besonderheiten

- Die VLAN-ID kann von 1 bis max 4094 gehen.

- Es gibt verschiedene VLAN-Tags
- IEEE 802.1Q (0x8100), dieses Tag ist der Normalfall.
- IEEE 802.1ad (0x9200, 0x9100)

1.2.1. IPv4 Adressnotation

- Eine IP-Adresse hat 4 Bytes.
- Sie wird normalerweise dezimal geschrieben (z.B. 194.45.71.113).

1.2.1. IPv4 Adressnotation

- Ein IP-Netz hat eine Netzwerk-Adresse
- Diese ist immer die erste IP-Adresse des Netzes (gerade).

- Ein IP-Netz hat eine Broadcast-Adresse
- Diese ist immer die letzte IP-Adresse des Netzes (ungerade).

1.2.2. IPv4 besondere Adressnotationen

Statt der dezimalen Schreibweise gibt es noch viele Varianten eine IP-Adresse zu schreiben:

- normal mit Punkten: 194.45.71.113
- als dezimale Ganzzahl: 3257747313
- hexadezimal mit Punkten: 0xC2.0x2D.0x47.0x71
- als hexadezimale Zahl: 0xC22D4771
- octal mit Punkten: 0624.0105.0161.0423
- als octale Ganzzahl: 03112735071423

1.2.3. IPv4 Netzmaske, Schreibweisen, Verfahren

Die Netzmaske gibt an, wie viele IP-Adressen in einen Netz sind.

Beispiele:

- 256 IP's bits /24 dez 255.255.255.000 hex 0xFFFFFF00
- 128 IP's bits /25 dez 255.255.255.128 hex 0xFFFFFF80
- 064 IP's bits /26 dez 255.255.255.192 hex 0xFFFFFFC0
- 032 IP's bits /27 dez 255.255.255.224 hex 0xFFFFFFE0
- 016 IP's bits /28 dez 255.255.255.240 hex 0xFFFFFFF0
- 008 IP's bits /29 dez 255.255.255.248 hex 0xFFFFFFF8
- 004 IP's bits /30 dez 255.255.255.252 hex 0xFFFFFFFC
- 002 IP's bits /31 dez 255.255.255.254 hex 0xFFFFFFFE
- 001 IP's bits /32 dez 255.255.255.255 hex 0xFFFFFFFF

1.2.4. IPv4 Adressbereiche

- Private Adressen, nicht im Internet gültig (RFC1918)
  - 10.0.0.0/8
  - 172.16.0.0/12
  - 192.168.0.0/16
- Multicast-Adressen
  - 224.0.0.0/4
- Loopback-Adressen
  - 127.0.0.0/8

1.3.1. IPv6 Wissenswertes

- IPv6-Adressen sind 16 Bytes (128 Bit) lang.
- Sehr großer Adressraum
- Ende-zu-Ende-Prinzip, kein NAT notwendig.
- Einfache Umnummerierung
- Unterstützung mehrerer Internetverbindungen gleichzeitig.

1.3.2. IPv6 Adressnotation

- IPv6 Adressen werden hexadezimal in 8 Blöcken zu je 16 Bit geschrieben.
- Diese Blöcke werden mit einem Doppelpunkt getrennt.
- Führende Nullen dürfen weggelassen werden.
- Die größte Reihe von Blöcken mit dem Wert Null wird durch 2 Doppelpunkte dargestellt.

1.3.2. IPv6 Adressnotation

- fe80::21e:ecff:fe9c:8887 == fe80:0:0:0:21e:ecff:fe9c:8887
- 2001:db8:100:6061::14 == 2001:db8:100:6061:0:0:0:0:14

- In einer URL wird die IPv6-Adresse in eckigen Klammern geschrieben:
- http://[2001:db8:100:6061::17]/

1.3.3. IPv6 Subnetze

- Subnetze werden mit der sogenannten "Prefixlen" festgelegt. Diese wird nach einem Schrägstrich der IPv6-Adresse angefügt.
- Das kleinste Netzwerk auf einem Ethernet ist ein "/64", d.h. 64 Bit stehen für die Geräte Zur Verfügung.

1.3.3. IPv6 Subnetze

- Ein Kunde sollte von seinem Provider mindestens ein "/56" bekommen, damit kann der Kunde 8 Bit für bis zu 256 Subnetze für sich verwenden.
- Üblich sind Zuweisungen bis zu "/48", damit kann der Kunde 16 Bit für bis zu 65536 Subnetze verwenden.

1.3.4. IPv6 Adressbereiche (Teil 1)

- Link-Local-Adressen, diese Adressen sind nur innerhalb des lokalen Ethernets gültig.

  fe80::/64

- Loopback Adressen

  ::1/128

- Multicast Adressen

  ff00::/8

1.3.5. IPv6 Adressbereiche (Teil 2)

- Unique-Local Adressen, privat, nicht zum Gebrauch im Internet.

  fc00::/7

- Global Unicast, dies sind normale öffentliche IP-Adressen.

  2000::/3

1.3.5. IPv6 Adressbereiche (Teil 2)

- IPv4 mapped, reserviert um IPv4 Adressen zu erreichen.

  ::ffff:0:0/96

- NAT64

  64:ff9b::/96

1.4.1. IPv4 Address Resolution Protocol (ARP)

Wenn ein Paket gesendet werden soll, so fragt der Rechner per Broadcast, wer für eine IP-Adresse zuständig ist. Der angesprochen Rechner antwortet per Unicast und meldet damit seine Mac-Adresse. Damit kann der Rechner das Datenpaket direkt an die Netzwerkkarte des Ziels senden. Diese gelernte Mac-Adresse wird dann für ca. 1 Minute vom Rechner gemerkt.

- Daher dauert das Übertragen des ersten Paketes etwas länger.

1.4.2. IPv4 Diagnose mit dem "arp" Befehl

Arp ist das wichtigste Kommando zur Anzeige was auf der untersten Ebene passiert. Es zeigt die gelernten Mac-Adresse eines Rechners.

1.4.2. IPv4 Diagnose mit dem "arp" Befehl

```
$ arp -a
gw2.dinoex.sub.de (194.45.71.1) at 00:15:0c:58:56:b3 on bge0 [ethernet]
uucp.dinoex.sub.de (194.45.71.2) at 00:04:76:dc:29:0e on bge0 [ethernet]
fw.dinoex.sub.de (194.45.71.4) at 00:00:24:c8:74:50 on bge0 [ethernet]
t4.dinoex.sub.de (194.45.71.6) at 00:19:db:f2:ca:40 on bge0 [ethernet]
t3.dinoex.sub.de (194.45.71.7) at 00:19:db:60:8c:2d on bge0 [ethernet]
t2.dinoex.sub.de (194.45.71.9) at 00:04:23:bb:e7:3b on bge0 [ethernet]
home3.dinoex.sub.de (194.45.71.20) at 00:18:f3:e2:eb:82 on bge0 [ethernet]
nppp40.lan-ks.de (194.45.71.240) at (incomplete) on bge0 [ethernet]
```

1.4.3. IPv4 Diagnose mit dem "ping" Befehl

- Mit dem Ping Kommando prüft man die Erreichbarkeit eines anderen Rechners. Auch wenn ein Firewall die Antwort verhindert, so lernt der sendende Rechner die Mac-Adresse des Ziels, falls es am gleichen Ethernet angeschlossen ist.

- Kann ARP keine Mac-Adresse ermitteln (00:00:00:00:00:00), so sollte man die Netzwerkkarte, die Kabel und Switches untersuchen.

1.5.1 IPv6 Neighbor Discovery Protocol (NDP)

Das Neighbor Discovery Protocol (NDP) ermittelt zu der gesuchten IPv6 Adresse die aktuelle MAC-Adresse.

1.5.1 IPv6 Neighbor Discovery Protocol (NDP)

```
$ ndp -a -n
Neighbor                                Linklayer Address  Netif Expire    S Flags
2001:db8:5001:1::20                     00:18:fe:62:3f:2b    em0 23h53m2s  S
2001:db8:5001:1::41                     00:19:db:f4:7f:c1    em0 22h43m17s S
2001:db8:5001:1::1                      00:00:24:c8:74:50    em0 23h59m58s S R
2001:db8:5001:1::2                      00:19:db:f4:7f:c1    em0 expired   D
2001:db8:5001:1::62                     00:30:48:cf:bc:d0    em0 permanent R
2001:db8:5001:1::43                     00:19:db:f4:7f:c1    em0 expired   D
2001:db8:5001:1::4                      00:19:db:f4:7f:c1    em0 23h59m48s S
2001:db8:5001:1::5                      00:30:48:cf:bc:d0    em0 permanent R
2001:db8:5001:1::6                      00:25:90:28:b7:ac    em0 17h33m38s S
2001:db8:5001:1::48                     00:30:67:c2:61:6f    em0 23h22m35s S
2001:db8:5001:1::50                     00:17:a4:fb:d8:04    em0 26s       R
2001:db8:5001:1::53                     00:30:48:cf:bc:d0    em0 permanent R
2001:db8:5001:1::35                     00:30:48:cf:bc:d0    em0 permanent R
2001:db8:5001:1::36                     00:30:48:cf:bc:d0    em0 permanent R
fe80::230:48ff:fecf:bcd0%em0            00:30:48:cf:bc:d0    em0 permanent R
```

1.5.2. IPv6 Autokonfiguration (RA)

- Ein IPv6 Router kann ein sogenanntes "Router Advertisement" senden.
- Damit kann ein Gerät über das Neighbor Discovery Protocols (NDP) vollautomatisch die Router im Netzwerk erkennen und die globalen Adresse-Prefixe erhalten. Über die eigenen MAC-Adresse wird dann eine vollständige IPv6-Adresse gebildet.

1.5.2. IPv6 Autokonfiguration (RA)

- Aus Datenschutzgründen gibt es die sogenannten "Privacy Extensions", diese verhindern, dass das Gerät eindeutig anhand seiner IPv6-Adresse identifiziert werden kann.

1.5.3. IPv6 Router Solicitation

- Ein IPv6 Rechner kann auf dem Ethernet mit einem Paket an die Multicast-Adresse

  FF02::2%if

  aktiv die aktuellen Router im Netz finden.

1.5.4. DHCPv6

- Da ein Gerät meist auch DNS-Server und weitere Daten braucht, werden IPv6-Adressen bevorzugt über DHCPv6 zugewiesen.

1.5.5. IPv6 Multihoming

- Es ist zulässig mehrere Router zu unterschiedlichen Providern zu haben.
- Dabei hat das Gerät im Netz gleichzeitig mehrere globale IPv6-Adressen.

- Das passiert z.B. wenn ein DSL-Router vom Provider einen neuen globalen Prefix zugewiesen bekommt.

1.5.6. IPv6 Path MTU Discovery

- Die Übertragung von Paketen mit einer MTU von 1280 Bytes ist in allen IPv6 Netzen garantiert.
- Damit auch größere Pakete übertragen werden können, muss die Path MTU Discovery (PMTU) auf allen Geräten funktionieren, d.h. ICMPv6 darf nicht blockiert werden.

1.6.1. Dual-Stack

- Ein Rechner mit "Dual-Stack" hat außer den IPv6 Adressen auch noch mindestens eine IPv4 Adresse.

1.6.2. Dual-Stack-Lite

- Bei "Dual-Stack Lite" hat der Rechner keine öffentliche IPv4 Adresse, sondern seine IPv4 Pakete werden durch IPv6 getunnelt und erst beim Provider zu einer öffentliche IPv4 Adresse zugewiesen.
- Dort findet die Übersetzung für alle Kunden statt. Dieser Router wird als "Carrier-Grade-NAT" oder kurz "CGN" bezeichnet.

1.6.3. IPv6 Only

- Ein IPv6 Rechner kann keine Verbindung mit einem IPv4 Rechner aufbauen.
- Aber mit NAT64 und DNS64 kann ein Dual-Stack Router dieses ermöglichen
- Leider muss der Router dafür originalen DNS-Daten manipulieren.

1.7. Gegenüberstellung

	IPv4	IPv6
Größe der Adressen	4 Byte	16 Byte
Schreibweise Adressen	Dezimal	Hexadezimal
Nachbarn	ARP	NDP
Autoconfiguration	DHCP	DHCPv6, RA
MTU	68 bis 1500	1280 bis 1500
Fragmentierung auf dem Weg	Ja	Nein

2. Protokolle

- 2.1. UDP (User Datagram Protocol)
- 2.2. TCP (Transmission Control Protocol)
- 2.3. ICMP (Internet Control Message Protocol)
- 2.4. HTTP (Hypertext Transfer Protocol)
- 2.5. SMTP (Simple Mail Transfer Protocol)
- 2.6. DNS (Domain Name System)

2.1. UDP (User Datagram Protocol)

- Die Pakete werden einfach gesendet. Bei Störungen muss das Programm die Daten erneut senden.

2.1. UDP (User Datagram Protocol)

- DNS (Domain Name System)

  Umsetzung eines Rechnernamens in eine IP-Adresse.
- NTP (Network Time Protocol)

  Synchronisieren von Datum und Uhrzeit.
- DHCP (Dynamic Host Configuration Protocol)

  Zentrale automatische Konfiguration der Rechner.

2.2.1. TCP (Transmission Control Protocol)

- Die Daten werden Verbindungs-orientiert gesendet. TCP kümmert sich bei Störungen automatisch und das erneute Versenden von Paketen, die verloren gegangen sind.
- sehr zuverlässig.
- Timeout groß, typisch ca. 2 Minuten.

2.2.1. TCP (Transmission Control Protocol)

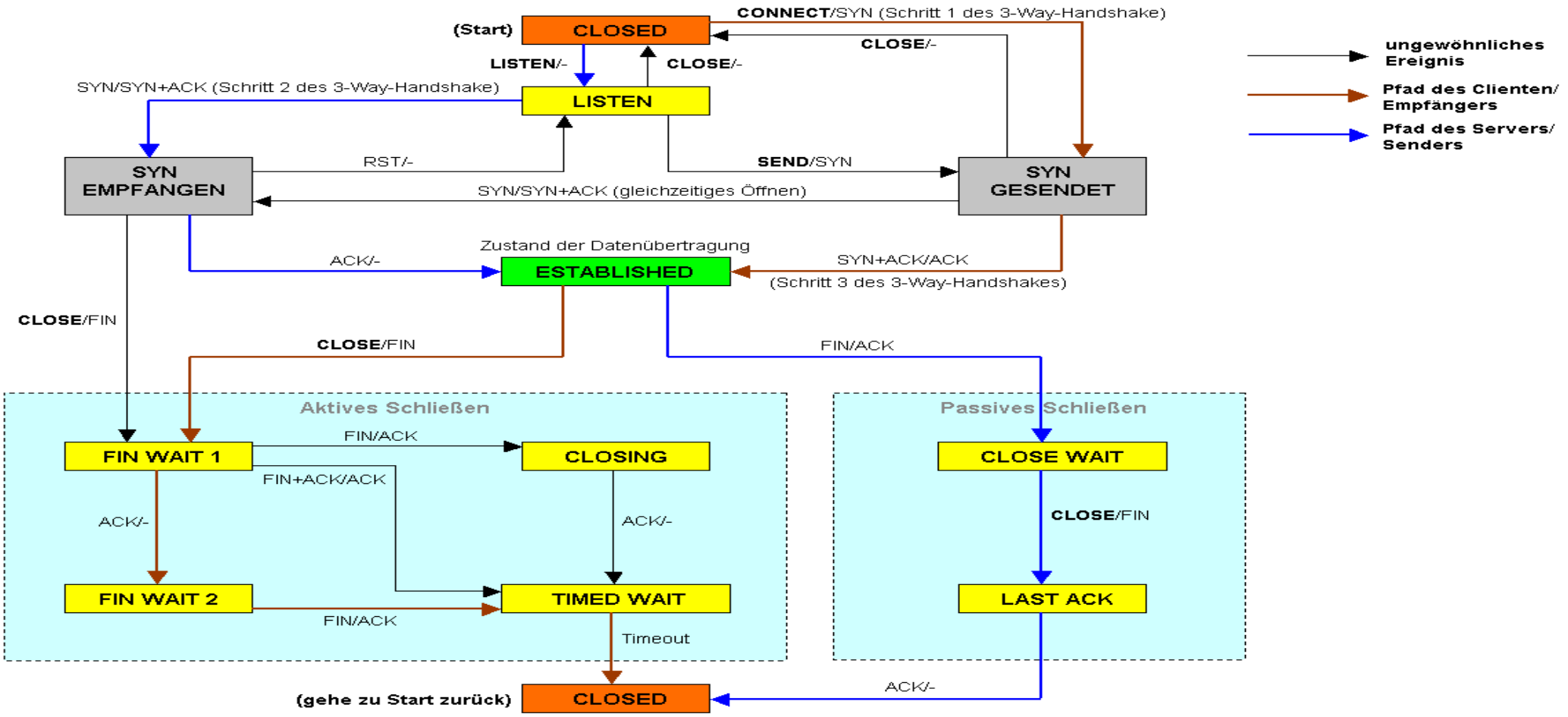
- Es braucht nur das verlorene Paket erneut gesendet werden, TCP gibt die Daten in der richtigen Reihenfolge an das Programm. Die Daten werden automatisch in passende Häppchen zerlegt und wieder zusammengesetzt. Die Datenmenge wird ebenfalls optimiert und passende der Übertragung gesteuert.

2.2.2. TCP Three-Way-Handshake

Beim Verbindungsaufbau finden folgende Schritte statt:

- SYN-Paket wird gesendet. Client ist im Status "SYN-SENT".
- Der Server liest das Paket und geht in den Status "SYN-RECEIVED". Er sendet ein SYN-ACK Paket zur Bestätigung der Verbindung.
- Der Client liest das Paket und sendet ebenfalls ein "SYN-ACK" Paket, damit ist die Verbindung erfolgreich aufgebaut, beide sind dann im Status "ESTABLISHED".

2.2.3. TCP Zustandsdiagramm



Quelle: http://de.wikipedia.org/wiki/Transmission_Control_Protocol

2.2.4. Diagnose TCP/IP

- Zur Diagnose gibt es das Kommando "netstat".
- Auf BSD Systemen außerdem "sockstat"
- Oder das Werkzeug "lsof" installieren.

2.2.4. Diagnose TCP/IP

```
$ netstat -a | less
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0 book64.50835           mail.dithmar-wes.telne SYN_SENT
tcp4       0     48 book64.ssh             home3.56439            ESTABLISHED
tcp4       0      0 book64.634             t3.nfsd                ESTABLISHED
tcp4       0      0 book64.1011            t2.nfsd                ESTABLISHED
tcp4       0      0 book64.954             t2.nfsd                ESTABLISHED
tcp4       0      0 book64.600             t3.nfsd                ESTABLISHED
tcp4       0      0 book64.860             t4.nfsd                ESTABLISHED
tcp4       0      0 book64.726             t4.nfsd                ESTABLISHED
tcp4       0      0 *.x11                  *.*                    LISTEN
tcp6       0      0 *.x11                  *.*                    LISTEN
tcp4       0      0 *.gdomap               *.*                    LISTEN
tcp4       0      0 *.ssh                  *.*                    LISTEN
tcp6       0      0 *.ssh                  *.*                    LISTEN
udp4       0      0 book64.673             t4.nfsd
udp4       0      0 book64.832             t4.nfsd
udp4       0      0 *.gdomap               *.*
udp4       0      0 *.syslog               *.*
udp6       0      0 *.syslog               *.*
icm4       0      0 *.*                    *.*
```

2.2.5. TCP Window-Size

In einem TCP-Paket wird der Gegenstelle mitgeteilt wie viel Platz in Speicher für die laufende Verbindung frei ist. Diese "Window-Size" wird dynamisch angepasst, und damit wird die effektive Bandbreite der Datenübertragung gesteuert.

2.3.1. ICMP (Internet Control Message Protocol)

- Das ICMP Protokoll ist notwendig für die Funktion von IP. Keinesfalls sollte man alle ICMP Pakete ausfiltern.
- ICMPv4 benötigt mind. Pakete mit den Typen 3 (Destination unreachable), 4 (Source Quench) und 11 (Time Exccced)
- ICMPv6 benötigt mind. Pakete mit den Typen 1 (Destination unreachable), 2 (Too Big), 3 (Time Exccced), 135 (Neighbor Solicitation) und 136 (Neighbor Advertisement)

2.4. HTTP (Hypertext Transfer Protocol)

```
$ telnet www.dwaz.de 80
Trying 80.237.237.55...
Connected to www.dwaz.de.
Escape character is '^]'.
GET /
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>403 Forbidden</TITLE>
</HEAD><BODY>
<H1>Forbidden</H1>
You don't have permission to access /
on this server.<P>
<HR>
<ADDRESS>Apache/1.3.41 Server at 80.237.237.55 Port 80</ADDRESS>
</BODY></HTML>
Connection closed by foreign host.
```

2.5. SMTP (Simple Mail Transfer Protocol)

```
$ telnet smtp.dinoex.de 25
Trying 188.40.204.4...
Connected to smtp.dinoex.de.
Escape character is '^]'.
220 smtp.dinoex.de ESMTP Sendmail 8.14.5/8.14.5; Fri, 4 Oct 2013 12:11:57
+0200 (CEST); (No UCE/UBE) logging access from: book64.dinoex.sub.de(OK)-
book64.dinoex.sub.de [194.45.71.113]
HELO me
250 smtp.dinoex.de Hello book64.dinoex.sub.de [194.45.71.113], pleased to
meet you
MAIL FROM: <dirk.meyer@dinoex.sub.org>
250 2.1.0 dirk.meyer@dinoex.sub.org... Sender ok
RCPT TO: <webmaster@fan-sub.de>
250 2.1.5 webmaster@fan-sub.de... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
```

2.5. SMTP (Simple Mail Transfer Protocol)

```
DATA
354 Enter mail, end with "." on a line by itself
Subject: test

Nur ein Test.
.
250 2.0.0 n4P4NLcn058704 Message accepted for delivery
QUIT
221 2.0.0 smtp.dinoex.de closing connection
Connection closed by foreign host.
```

2.6.1. IPv4 DNS (Domain Name System)

- IPv4 Hosts werden im DNS als Einträge vom Typ "A" der Zone hinzugefügt.

  Beispiel:

```
smtp.dinoex.de.          IN      A       188.40.204.4
```

2.6.1. IPv4 DNS (Domain Name System)

- Beim sogenannten "Reverse" Eintrag wird die IPv4 Adresse in die 4 dezimalen Oktets zerlegt und rückwärts der Zone "in-addr.arpa" zugewiesen.

  Aus 194.45.71.113 wird also:

  113.71.45.194.in-addr.arpa.

  Beispiel:

```
113.71.45.194.in-addr.arpa. IN PTR book64.dinoex.sub.de.
```

2.6.2. IPv6 DNS (Domain Name System)

- IPv6 Hosts werden im DNS als Einträge vom Typ "AAAA" der Zone hinzugefügt.
- Ein Host kann sowohl IPv4 als auch IPv6 Adressen haben. Die IPv6 Adresse wird bevorzugt. Beispiel:

```
smtp.dinoex.de.          IN      A       188.40.204.4
smtp.dinoex.de.          IN      AAAA    2a01:4f8:100:6061::14
```

2.6.2. IPv6 DNS (Domain Name System)

- Beim sogenannten "Reverse" Eintrag wird die IPv6 Adresse in alle Hex-Digits zerlegt und rückwärts der Zone "ip6.arpa" zugewiesen. Die Nullen dürfen hier nicht weggelassen werden. Aus 2a01:4f8:100:6061::14 wird also:

```
4.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.6.0.6.0.0.1.0.8.f
.4.0.1.0.a.2.ip6.arpa. IN PTR smtp.dinoex.de.
```

3. Sicherheit

- 3.1. Technik
- 3.2. Network Address Translation (NAT)
- 3.3. Fragementierung
- 3.4. Portscans
- 3.5. Neue Angriffe
- 3.6. Applikationen und Tunnel

3.1. Technik

- Firewall

  Ein Router, der die Verbindungen auf Paketebene blockiert oder zulässt.

  Verbindungen werden typischerweise per NAT (Network Address Translation) nach außen weitergegeben.

3.1. Technik

- Proxy

  Ein Rechner der die Verbindungen annimmt, und die Daten weiterreicht. Es gibt keine direkte Verbindung zwischen dem Rechner des Benutzers und dem externen Server.

3.2. Network Address Translation (NAT)

NAT wird in der IPv4-Welt fast überall eingesetzt. Duch NAT sind die Rechner erst mal nicht von außen erreichbar. Mit IPv6 muss man im Firewall alle eingehenden Verbindungen filtern.

3.3. Fragementierung

- In IPv4 können Pakete unterwegs beliebig fragmentiert werden. Es konnten fragmentierte Pakete dazu verwendet werden Filter in Firewalls auszutricksen.

- In IPv6 werden Pakete unterwegs nicht fragmentiert. Hier ist es aber wichtig das der Sender die max. Paketgröße mit PMTU erkennen kann.

3.4. Portscans

- In IPv6 sind die Netze zu groß um einen Portscan in einem Subnetz zu machen, der Adressraum ist 64 Bit breit.
- Eine einzelne IP-Adresse lässt sich wie bisher abtasten.
- Aber in IPv6 kann man alle Knoten mit der Multicast-Adresse ff02::1 erreichen.
- Rechner kann man auch über DNS, Suchmaschinene und andere Quellen finden.

3.5. Neue Angriffe

- ND Spoofing (früher ARP Spoofing)
- Router Advertisement Spoofing
- NDP sollte in offenen Netzen gefiltert werden:
  - ICMP-Type 133 bis 137
  - Quelladresse ist in ::/128 oder fe80::/16
  - Zieladresse ist in fe80::/16 oder ff02::/16
  - Hop-Limit muss 255 sein.

3.6. Applikationen und Tunnel

- Die Zugangslisten (ACL) am Server gelten für IPv4 und IPv6. Hier darf man kein Protokoll vergessen.

- Es gibt viele Möglichkeiten einen IPv6 Tunnel aufzubauen und damit den IPv4 Firewalls zu umgehen, z.B. TEREDO und 6to4.

4. Programmierung

- 4.1. TCP Server
- 4.2. TCP Client
- 4.3. TCP Test
- 4.4. SSL Zertifikat
- 4.5. SSL Server
- 4.6. SSL Client
- 4.7. SSL Test
- 4.8. Dual-Stack Server

4.1. TCP Server

- Für IPv6 wird der Server statt an "0.0.0.0" an "::" gebunden.

  Hier ein Beispiel in Ruby:

4.1. TCP Server

```
require 'socket'
require 'thread'

def show_addr(addr)
  family, port, _host, ip_address = addr
  "#{family} on ip_address=#{ip_address} port=#{port}"
end
```

4.1. TCP Server

```
server = TCPServer.new '::', 8080
put 'Server ' + show_addr(server.addr)
loop do
  # Wait for a clients to connect
  Thread.fork(server.accept) do |client|
    puts 'accept local ' + show_addr(client.addr)
    puts 'accept remote ' + show_addr(client.peeraddr)
    client.puts 'Hello !'
    client.puts "Time is #{Time.now}"
    client.close
  end
end
```

4.2. TCP Client

- Im Client sind in der Regel keine Änderungen erforderlich.

  Hier ein Beispiel in Ruby:

4.2. TCP Client

```
require 'socket'

def show_addr(addr)
  family, port, _host, ip_address = addr
  "#{family} on ip_address=#{ip_address} port=#{port}"
end

client = TCPSocket.new 'b3.example.com', 8080
puts 'connect local ' + show_addr(client.addr)
puts 'connect remote ' + show_addr(client.peeraddr)
while line = client.gets
  puts "received : #{line.chop}"
end
client.close
```

4.3.1. TCP Test Telnet

- Wir starten den Server:
  $ ./server1.rb
  Server AF_INET6 on ip_address=:: port=8080

4.3.1. TCP Test Telnet

- Ein erster Test mit "telnet":
  $ telnet b3.example.com 8080
  Trying 2001:db8:5001:1::103...
  Connected to b3.example.com.
  Escape character is '^]'.
  Hello !
  Time is 2020-02-27 12:14:15 +0100
  Connection closed by foreign host.

- Auf dem Server:
  accept local AF_INET6 on ip_address=2001:db8:5001:1::103 port=8080
  accept remote AF_INET6 on ip_address=2001:db8:221:441::2 port=24128

4.3.2. TCP Test Ruby

- Der Test mit dem Ruby Programm:
  $ ./client1.rb
  connect local AF_INET6 on ip_address=2001:db8:5001:1::103 port=16461
  connect remote AF_INET6 on ip_address=2001:db8:5001:1::103 port=8080
  received : Hello !
  received : Time is 2020-02-27 12:19:16 +0100

- Auf dem Server:
  accept local AF_INET6 on ip_address=2001:db8:5001:1::103 port=8080
  accept remote AF_INET6 on ip_address=2001:db8:5001:1::103 port=16461

4.4.1. SSL Certificate Authority

- Für SSL brauchen wir ein signiertes Zertifikat.
  In unserem Beispiel machen wir eine eigenen CA.

```
mkdir demoCA
cd demoCA
mkdir certs newcerts private
chmod 700 private
echo "01" > serial
touch index.txt

openssl req -new -x509 -keyout private/cakey.pem -out cacert.pem
[..]
Common Name (e.g. server FQDN or YOUR name) []:ca.example.com
[..]
```

4.4.2. SSL Zertifikat

- Wir erstellen den Private-Key und den Certificate-Signing-Request:

```
openssl req -nodes -new -keyout private/key.pem -out certs/csr.pem
[..]
Common Name (e.g. server FQDN or YOUR name) []:b3.example.com
[..]
```

- Oder komplizierter mit Subject Alternative Name:

```
openssl req -nodes -new -addext "basicConstraints=CA:FALSE" \
    -addext "extendedKeyUsage=serverAuth" \
    -addext "keyUsage=nonRepudiation,digitalSignature,keyEncipherment" \
    -addext "subjectAltName=DNS:b3.example.com" \
    -keyout private/key.pem -out certs/csr.pem
```

4.4.2. SSL Zertifikat

- Jetzt können wir den CSR mit der CA signieren:

```
cd ..
openssl ca -policy policy_anything -out demoCA/certs/cert.pem -infiles
demoCA/certs/csr.pem
```

- Oder komplizierter mit Subject Alternative Name:

```
cat > extensions.cnf << EOF
basicConstraints=CA:FALSE
keyUsage=nonRepudiation,digitalSignature,keyEncipherment
extendedKeyUsage=serverAuth
subjectAltName=DNS:b3.example.com
EOF
cd ..
openssl ca -policy policy_anything -extfile demoCA/extensions.cnf -out
demoCA/certs/cert.pem -infiles demoCA/certs/csr.pem
```

4.5. SSL Server

- Der Server benötigt nur sein eigenes Zertifikat und den Private-Key.
- Bei einer offiziellen Zertifizierungsstelle braucht der Server auch alle Zwischen-Zertifikate.

- Im Ruby Programm müssen wir jetzt die SSL Fehler abfangen, damit der Server weiter laufen kann.

4.5.1. SSL Server (Teil 1)

```
require 'socket'
require 'thread'
require 'openssl'

CERT_FILE = 'demoCA/certs/cert.pem'.freeze
KEY_FILE = 'demoCA/private/key.pem'.freeze

def show_addr(addr)
  family, port, _host, ip_address = addr
  "#{family} on ip_address=#{ip_address} port=#{port}"
end
```

4.5.2. SSL Server (Teil 2)

```
tcp_server = TCPServer.new '::', 8443 # Server bind to port 8443
puts 'Server ' + show_addr(tcp_server.addr)
context = OpenSSL::SSL::SSLContext.new
context.cert = OpenSSL::X509::Certificate.new File.open(CERT_FILE)
context.key = OpenSSL::PKey::RSA.new File.open(KEY_FILE)
ssl_server = OpenSSL::SSL::SSLServer.new(tcp_server, context)
```

4.5.3. SSL Server (Teil 3)

```
loop do
  begin
    # Wait for a clients to connect
    Thread.fork(ssl_server.accept) do |client|
      puts 'accept local ' + show_addr(client.addr)
      puts 'accept remote ' + show_addr(client.peeraddr)
      client.puts 'Hello !'
      client.puts "Time is #{Time.now}"
      client.close
    end
  rescue
    # report error and continue
    $stderr.puts $!
  end
end
```

4.6 SSL Client

- Der Client braucht nur das öffentliche Zertifikat der CA.
- Bei einer offiziellen Zertifizierungsstelle braucht man es nicht angeben.

4.6.1. SSL Client (Teil 1)

```
require 'socket'
require 'openssl'

CA_FILE = 'demoCA/cacert.pem'.freeze

def show_addr(addr)
  family, port, _host, ip_address = addr
  "#{family} on ip_address=#{ip_address} port=#{port}"
end
```

4.6.2. SSL Client (Teil 2)

```
tcp_client = TCPSocket.new 'b3.example.com', 8443
puts 'connect local ' + show_addr(tcp_client.addr)
puts 'connect remote ' + show_addr(tcp_client.peeraddr)
context = OpenSSL::SSL::SSLContext.new
context.verify_mode = OpenSSL::SSL::VERIFY_PEER |
                      OpenSSL::SSL::VERIFY_FAIL_IF_NO_PEER_CERT
context.ca_file = CA_FILE
ssl_client = OpenSSL::SSL::SSLSocket.new tcp_client, context
ssl_client.connect
while line = ssl_client.gets
  puts "received : #{line.chop}"
end
ssl_client.close
```

4.7.1. SSL Test OpenSSL

- Wir starten den Server:
  $ ./server2.rb
  Server AF_INET6 on ip_address=:: port=8443

4.7.1. SSL Test OpenSSL

- Auf dem Client:
  $ openssl s_client -connect b3.example.com:8443 -Cafile demoCA/cacert.pem
  [...]
  Hello !
  Time is 2020-02-27 13:16:35 +0100
  closed

- Auf dem Server:
  accept local AF_INET6 on ip_address=2001:db8:5001:1::103 port=8443
  accept remote AF_INET6 on ip_address=2001:db8:221:441::2 port=49015

4.7.2. SSL Test Ruby

- Auf dem Client:
  $ ./client2.rb
  connect local AF_INET6 on ip_address=2001:db8:5001:1::103 port=41191
  connect remote AF_INET6 on ip_address=2001:db8:5001:1::103 port=8443
  received : Hello !
  received : Time is 2020-02-27 13:24:56 +0100

- Auf dem Server:
  accept local AF_INET6 on ip_address=2001:db8:5001:1::103 port=8443
  accept remote AF_INET6 on ip_address=2001:db8:5001:1::103 port=41191

4.8. Dual-Stack Server

- Ein Dual-Stack Server muss für IPv4 und IPv6 jeweils einen eigenen Socket benutzen.

  Beispiel:

  https://github.com/ruby/ruby/blob/master/sample/dualstack-httpd.rb

5.1. Links

- http://www.benedikt-stockebrand.de/ipv6-workshop_de.html
- http://www.benedikt-stockebrand.de/resources_de.html
- http://events.ccc.de/congress/2010/Fahrplan/attachments/1808_vh_thc-recent_advances_in_ipv6_insecurities.pdf
- DFN-Cert IPv6 und die Sicherheit vom 13.03.2013.

5.2. Fragen und Feedback

- Fragen
- Feedback