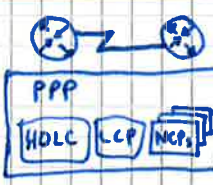
dnet 2

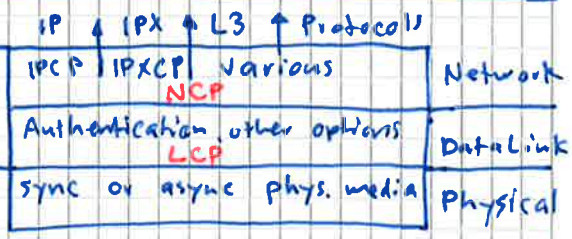
PPP

NCP : Network Control Protocol

PPP : Point-to-Point Protocol
baut auf HDLC auf
RFC 1547, L2

LCP : Link Control Protocol





Verbindungsaufbau PPP

1. Link Establishement
2. Kontrolle Link Qualität
3. Aushandlung L3 Optionen

```
(conf-if)# enc ppp
(conf-if)# compr [stack | predictor]
(conf-if)# ppp quality [0..100]
```

```
#sh int se 0/0
# debug ppp
```

Authentifizierung mit PPP

PAP: Password Authentication Protocol
- 2-way handshake
- sendet Passwort in Klartext → unsicher

CHAP: Challenge handshake authentication p.
- 3 way handshake (mit Challenge)

```
(conf)# username USER password PW
(conf)# int se 0/0
(conf-if)# ppp authentication chap
(conf)# debug ppp authentication
#aaa
```

FR

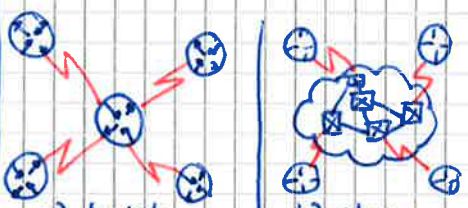
FR: Frame Relay
- abgespeckte Version von X.25
- Fehlerkorrektur auf L4
- verbindungsor.

VC : Virtual Circuit
- PVC: permanent

FRAD : Frame Relay Access Device
- Endgerät des Kunden in VC
- Rolle des DTE

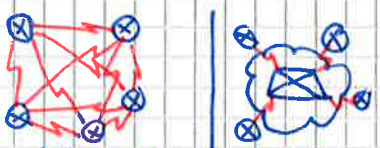
DLCI : Data Link Connection Identifier
- identifiziert VC
- nur lokal auf dem Link gültig

Hub + Spoke Topologie



⊕ günstiger da Kunde pro IF zahlt

vollvermaschte Topologie



Gründe für vollverm:
- ⊕ Server geogr. verteilt
- ⊕ hohe Verfügbarkeit zw. i-gend

LMI: Local Mgmt Int
- inverse Arp, d.h. MAC → IP
- dyn. Zuordnung DLCI ↔ IP

Probleme bei DV Routing-protokollen wegen split Horizon Regel
→ Subinterfaces

```
(conf)#int Fa 0/1
(conf-if)# encapsulation frame-relay ietf
(conf-if)# no frame-relay inverse-arp
(conf-if)# frame-relay map ip 10.1.1.2 102 broadcast //statisch mapping
```

```
# show frame-relay map
# show frame-relay lmi
# show frame-relay pvc 115
# debug frame-relay lmi
```

```
(conf)#int se 0/0/1
(conf-if)# no shutdown
(conf-if)# no ip address
(conf-if)# int se 0/0/1.112 point-to-point
(conf-if)# encapsulation frame-relay ietf
(conf-if)# ip address ...
(conf-if)# frame-relay interface-dlci 115
```

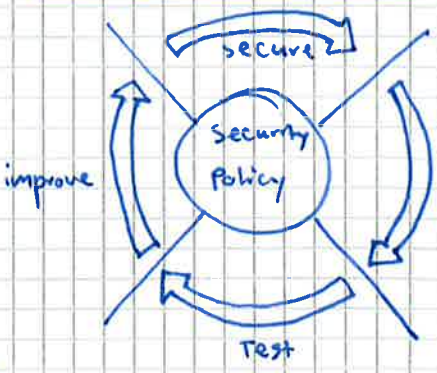
Network Security

Sicherheitsrichtlinien zwingend

Typen von Attacken

- Aufklärungsattacken
- Zugangsattacken
  - + Brute Force
  - + Ausnützen Vertrauensverh.
  - + Port Weiterleitung
  - + Man in the Middle
- DOS (Denial of Service)
  - + DDOS (Distributed ")
- Schadcode
  - + Viren - an anderes Prog. angeh.
  - + Würmer - Code ausf. und sich auf Syst.
  - + Trojaner - Informationsbesch.

Network Security Wheel



Kritische Dienste
- CDP
- TCP small servers
- UDP "
- Finger
- Http server
- bootp server
- auto-loading conf
- IP src routing
- proxy ARP
- IP directed BC
- classless Routing
- IP
- ver
- vc
- auto secure

IDS: Intrusion Detection System

SSH Zugang auf Router

```
(conf) line vty 0 4
(conf-li) no transport input
(conf-li) transport input ssh
(conf-li) login local

(conf) hostname ..
(conf) ip domain-name ...
(conf) crypto key generate rsa
(conf) username cisco secret cisco
[(conf) ip ssh time-out 15]
[(conf) ip ssh authentication-retries 2]
```

ungen. Leitungen sichern

```
(conf) line aux 0
(conf-l) no password
(conf-l) login
(conf-l) exit
```

Int passiv für Routing

```
(conf)# router rip
(conf-r)# passive-interface default
(conf-r)# no passive-interface ...
```

access-liste auf vty

```
(conf-l) access-class 2 [in|out]
(conf)# aaa new-model ?
```

ACL

ACL: Access control List
- muss einem IF zugeordnet sein
- max 1 ACL / L3 Protokoll, IF und Richtung (in/out)
- wirkt nur auf ein- und austretende, nicht jedoch auf vom Gerät generierte Pakete
- am Ende jeder Liste steht implizites "deny all"

→ ACL können nummeriert ODER benannt werden
Namen in Grossbuchstaben

→ einzelne Statements einer ACL können nicht gelöscht werden!

1. Standard ACL # 1...99, 1300...1999

- filtern aufgrund Source Adresse

```
(conf)# access-list 10 permit <IP> <Wildcard>
```

→ möglichst nahe zum Ziel definieren

2. Extended ACL # 100..199, 2000-2699

- filtern zusätzlich Destination Adr. + Port, Source Port, Protokoll

```
(conf)# access-list 10 permit tcp <IP> <Wildcard> any eq 80
                              (prot, source, dest, port)
```

→ möglichst nahe bei Quelle def.

① `(conf)# access-list number deny|permit|remark [source-addr [source-wildc.] [log] | host addr`

geht auch für vty z.B.:
```
(conf)# int Se0/0
(conf-if)# ip access-group 1 [in|out]

(conf)# ip access-list standard NAME
(conf-acl)# [51] deny host...
"  # permit IP wildcard
"  # int Fa0/0
(conf-if)# ip access-group NAME [in|out]

#sh access-lists
```

② `(conf)# access-list number deny|permit|remark protocol sourceAddr [sourceWC] [portNo|name] destAddr [operator] [portNo|name] [established]`

[operator]: lt, gt, eq, neq, range

[established]: Pakete gehen durch, falls TCP bereits steht

WC = Wildcard