Themen-Brainstorming Nachrichtentechnik

Klaus Kusche, Sept. 2026

Nachrichtentechnik:

Übertragungsmedien, Physik der Datenübertragung:

- Draht: Verdrillte Leitung, Koax oder Twinax
  - Verdrillung, Abschirmung (Sinn? Ab Cat 7 Doppelschirm Pflicht!)
  - Aufbau eines Cat-Kabels, Cat-Klassen, zugeordnete Ethernet-Speed
  - Unterschied von Cat 1-7 / Cat 8:
    Geschirmte RJ45-Stecker & -Buchsen oder neuere, inkompatible Stecker-Typen
  - Bei Ethernet: Streckenlänge max. 100 m (nur 5 + 90 + 5 !), Cat 8 @ 40 Gbit: 30 m
  - Abschluss-Widerstand (z.B. bei Koax-Busverkabelung): Gegen Signal-Reflexionen
    (auch bei Kurzschluss Reflexion)
    Funktionsprinzip bei Kabeltestern:
    Entfernung zu Bruchstelle / Kurzschluss durch Zeitmessung der Reflexion
- Elektr. Signal: Asymmetrisch, pseudo-symmetrisch oder symmetrisch = differentiell
  Sinn von differentiell!
- Signal auf Draht:
  Bei Ethernet direkt (siehe unten)
  Bei Kabelfernsehen QAM auf *einem hochfrequenten* Kanal
  Bei DSL und PowerLine QAM auf *vielen niederfrequenten* Kanälen
  → Je höher die Trägerfrequenz, umso höhere Datenrate möglich
  → Je höher die Datenrate bzw. QAM in einem Kanal, umso breiter der Kanal
  (d.h. umso mehr vom Frequenzbereich wird beansprucht)
- Licht: Freier Laser (heute selten, vor allem rechtliche Bedeutung) oder Glasfaser
  Glasfaser:
  - Reflexion an der Grenzschicht verschiedener optischer Dichten
    → Grenz-Biegedurchmesser
  - Multimode (kleinere Geschwindigkeit & Länge) / Singlemode
    (je nach Durchmesser 50-62,5 / 3-9 µm und Dichte-Profil)

  Höhere Datenrate, weniger Verluste → Reichweite einige km im LAN
  Wegen unterschiedlicher Dämpfung je nach Lichtfrequenz: IR-Licht 850-1650 nm
  → LED oder Laser-Diode
  → Sichtbares rotes Licht ist zusätzliches Sicherheitslicht, nicht Signalträger
- Glas im LAN: 10-400 Gbit (bis 100 Gbit/s monochrom, darüber 4-8 Wellenlängen)
  - Flexible Ankopplung mit wechselbarem GBIC / SFP, auch wegen Alterungs-Tausch
- Glas im Fernverkehr:
  - Viele Bänder / Trägerfrequenzen („Lichtfarben“) auf einer Ader

- ev. 2 Polarisationsrichtungen pro Kanal
  0,4 nm bzw. 50 GHz Kanalbreite, max. 100 Gbit/Band, 100-200 Bänder pro Ader
  ev. mehrere Bänder zu einem Kanal kombiniert → mehrere TB/s pro Ader!
- Max. Segment-Länge rund 100 km, dann optischer Verstärker
- Max. Streckenlänge rund 2000 km, dann elektrische Regeneration

- Funk:
  - Abhängig von Frequenz:
    – Datenrate
    – Reichweite
    – Durchdringung
    – „Geht nur auf direkte Sicht“ oder „Geht auch um’s Eck“
  - Extrembeispiele:
    – DCF-Sender (Zeit-Sender Frankfurt):
      Langwelle, einige 1000 km Reichweite, sehr geringe Bitrate
    – 27 MHz (CB-Funk, analog für Sprache)
    – Professioneller Daten-Richtfunk: Meist 60-75 GHz (genehmigungspflichtig)
    – Sonderfall Kurzwelle: Geht um die Welt, atmosphärische Reflexion

- Funk Sonderfälle:
  - UWB („Ultra Wide Band“):
    – Spezielle Funktechnik, keine Trägerfrequenz, sehr hohe Kanalbreite (einige GHz)
    – Ultrakurze Pulse, sehr wenig Energie pro Frequenz
      → stört andere angeblich nicht, lizenzfrei
    – Anwendungen:
      - Hohe Datenrate auf kurze Distanzen (Video, Bilddaten)
      - Positions-Triangulierung (teure Autoschlüssel, Apple Air Tags, Fabrikshallen)
  - LoRaWan („Long Range Wide Area Network“):
    – Niedrige Datenrate (einige kbit/s), 2-40 km Reichweite
    – Datenaustausch nicht ständig, Wechsel zwischen Pause und Aktivität
    – Ziel: minimaler Energieverbrauch bzw. maximale Effizienz
    – Geht durch Wände und Decken (400-1000 MHz)
    – Anwendung z.B. Fern-Zählerablesung (Wasser, Strom, …)
  - NFC/RFID: Extrem kurze Reichweite, Karten-tauglich, zugleich Stromversorgung
  - Bluetooth
  - Home-Automation-Funknetze: Zigbee, Z-Wave, KNX-RF, Homematic, …
  - DECT (Schnurlos-Telefone)

- Funk Regulierung:
  - Internationale Rahmenvereinbarung
  - Bundesnetzagentur (Frequenzplan und viele Verordnungen):
    – Frei oder lizenzpflichtig / kostenpflichtig?
    – Frequenzen, Kanalbreiten, Sendestärken, nur indoor?, Richtfunk erlaubt?, …
    – Auch Radar, Astronomie, …

- Linux: regulatory.db
  (verpflichtend, signiert, wird von allen WLAN-Treibern benutzt)

Datenübertragung:

- Topologie: Bus, Ring, Stern (Token Ring: Logisch Ring, physisch Stern)
- Medien-Zugriffs-Kontrolle, mehrere Übertragungen
  - Leitungsvermittlung, Frequenzmultiplex, Zeitmultiplex, Paketvermittlung
    Zeitmultiplex (aus der Telekom-Welt): ISDN, ATM, SDH/SONET
    ATM = Zeitmultiplex von Telefon und Daten mit fixen Zellen von 48+5 Bytes
    Die Backbone-Technologie von ~1980-2010, Betriebsende D-Telekom 2020
  - CSMA/CD, Halbduplex/Vollduplex, Token Ring,...
  - Deterministisch bzw. echtzeit-fähig???
    → Deterministische Ethernet-Varianten / Feldbusse (Profinet, Ethercat, Sercos, ...)
- Modulationsverfahren bei Funk, Docsis, DSL, teilweise auch Glas im Fernverkehr:
  AM, FM, PSK, QAM, meist QAM + Gray-Code
  „Modulation" bei Ethernet: PAM-3 bis PAM-16, keine klassische Träger-Welle
- Codierungsverfahren (8b10b-Code, 64b/66b-Code, 128b/130b-Code)
  - Rückgekoppelte Schieberegister
  - Sinn:
    – Gleichstromfreiheit, Galvanische Trennung (Trafo / Kondensator)
    – Takt-Synchronisation auf Empfängerseite (max. 5 gleiche Bits), mindest-Takt
    – Fehlererkennung
- Ethernet:
  T ... Twisted Pair = Cat (100 m bis 10 Gbit/s, 30 m bis 40 Gbit/s (unüblich))
  CR ... Twinax (7 m bis 100 Gbit/s)
  alles andere ... Glas (40 km bis 100 Gbit/s, 10 km bis 400 Gbit/s)
- „Altes" Ethernet (Thickwire, Thinwire, Cat-Kabel mit Hubs): CSMA/CD, Half-Duplex
  (zur Kollisionserkennung bekommt auch der Sender sein Signal gespiegelt)
  (bei Cat Host-to-Host ohne Hub: Eventuell Full Duplex)
  → Zusammenhang zwischen Mindest-Paketgröße und maximaler Netzausdehnung
  (wegen Laufzeit und Kollisionserkennung: Minimale Paket-Dauer muss
  länger sein als 2 x worst-case-Signallaufzeit, „RTDT" Round Trip Delay Time)
- Früher bei Cat-Kabel: Dedizierte Signal-Richtung für jedes Adernpaar
  → Bei direkter Verbindung zweier Endgeräte:
  Sendendes Adernpaar mit sendendem Adernpaar verbunden, ebenso Empfänger??
  → Unterscheidung 1:1-Kabel oder Kreuz-Kabel, separater Uplink-Stecker am Hub
  Heute: „Auto-MDI-X" (ab 100 Mbit optional, ab 1 Gbit Ethernet Pflicht):
  Auskreuzung und Speed wird automatisch ausgehandelt
  (Kreuzkabel noch immer bei serieller Schnittstelle = „Nullmodem")

- Ethernet heute (mit Switches, Routern):
  Keine Kollisionen mehr, kein CSMA/CD, Full Duplex, jeder kann jederzeit senden
- PoE: Power over Ethernet: Einspeisung einer Gleichspannung
  zwischen Signal-Adernpaaren (nicht einzelnen Adern!), ~48 V, < 1 A, max. 70 W

Netzwerke:

- Bedeutende LAN-Netze außer IP / Ethernet:
  - Infiniband für Supercomputer-Cluster und Cluster-SAN:
    meist Kupfer (oder Glas), lokal (<15 m), bis 800 Gbit/s
  - Fibre Channel (FC oder FCoE) für SAN („Storage Area Network“)
    Glas, auch über ein paar km, bis 64 Gbit/s
- OSI Modell:
  - Auf jeder Ebene abwärts:
    – Split / Merge der Nutzdaten (z.B. wenn größer als max. Paketlänge)
    – Header vorn dran
    – Ev. Prüfsumme hinten dran (Hardware / Schieberegister oder Software)
  … und aufwärts genau umgekehrt
  - „Gebirge-Modell“
- OSI-Modell & Hub/Switch/Router:
  Hub = Layer 1, regeneriert phys. Signal
  Switch = Layer 2, nach phys. Adresse (MAC)
  Router = Layer 3, nach log. Adresse (IP)
  Router (ev. auch Switch/Hub bei einheitlichen Mac-Adressen): Ev. Medien-Wechsel
  Router und Switches (Hubs normalerweise nicht): Geschwindigkeits-Umsetzung

  Routen:
  - Selbstlernend für lokale Geräte (und ARP/RARP)
  - Statisch in Tabellen eingetragen (ev. mit Gewichten / Prioritäten)
  - Über Routing-Protokolle (BGP, RIP oder OSPF) gelernt
    BGP („Border Gateway Protocol“): Ist ein EGP („Exterior Gateway Protocol“)
    = Global am Internet, nach AS (nicht IP-Adr), dynamisch (Sabotage-Gefahr!)
    RIP („Routing Information Protocol“): Ist ein IGP („Interior Gateway Protocol“)
    = Innerhalb eines AS, routet nach IP bzw. Subnetzen, diverse Schwächen
    OSPF („Open Shortest Path First“) &
    EIGRP („Enhanced Interior Gateway Routing Protocol“):
    Nachfolger von RIP

  Unterschied Hub und Switch:
  Hub: Jeder bekommt alles (logisch: Bus-Topologie), nur 10 & 100 Mbit Ethernet
  Switch: Jeder bekommt nur seine Daten (+ Broadcast)
  → Nur 1 Übertragung gleichzeitig pro Hub, viele pro Switch → Mehr Durchsatz!
  → Beim Hub kann jeder fremde Übertragungen mitlesen, beim Switch nicht

→ Switches arbeiten meist „Store and Forward“, selten „Cut-Through“
Hubs sind fast immer „Cut-Through“
- Switches haben mehr Verzögerung als Hubs
- Hubs haben kein Paket-Stau-Problem

Unterschied Layer-3 Switch und Router?
→ Router wird gezielt angesprochen mit eigener IP-Adresse, Switch nicht
→ Router trennt lokale Subnetze und Broadcast-Domänen,
Switch trennt logisch keine Netze und ist Broadcast-transparent

Problem bei Store&Forward-Switches und bei Routern,
besonders bei Geschwindigkeits-Umsetzung:
Paket-Stau am Switch / Router, Pakete werden verworfen
Router kann mit „ICMP Source Quench“ reagieren, Switch nicht
(aber ICMP Source Quench ist auch eine Denial-of-Service-Möglichkeit)
Sonst (nur bei TCP): Im Laufe der Zeit Verzögerung durch fehlende Ack's

Bridge & Repeater:
Bridge ist „Punkt-to-Punkt Layer 2 Verbindung“ (wie Switch):
„Transparente“ Bridge oder nicht?
ev. filternd: Nur Daten für den anderen Netz-Teil durchleiten (ev. selbstlernend)
ev. mit Trennung der Broadcast-Domänen
ev. mit Medien-Wechsel
Repeater ist „Punkt-to-Punkt Signal-Regenerierung“ (wie Hub)
„Repeater“, z.B. im WLAN oder Handy-Netzen:
Auch Signal-Umsetzung / -Weiterleitung:
Bei Store & Forward Empfangen & Senden im gleichen Band:
Halbierung des Durchsatzes!

- OSI Ebene 2, MAC-Adressen, ARP/RARP, ...
(auch auf Ebene 2 gibt es MAC-Broadcast!)
Hat Protokoll-Feld im / nach dem Header für das Protokoll auf Layer 3 (viele)

- Trunking = „gebündelte Verbindung“, mehrere Bedeutungen!
- LAN „Port Trunking“, „Etherchannel“, „Bonding“, „Teaming“, „Link Aggregation“:
Mehrere phys. Leitungen zwischen denselben Geräten
bilden eine logische Verbindung (schneller, Ausfall-sicherer)
Braucht spezielle Unterstützung bei MAC-Adressen-Tabelle und Broadcasts
- LAN „Trunk Port“: Mehrere logische (V)LAN's über eine phys. Leitung
- Telekom „SIP Trunk“:
Mehrere virtuelle „analoge“ Telefonverbindungen über eine Leitung

- STP „Spanning Tree Protokoll“: Erkennt Schleifen in der LAN-Verkabelung
→ Schaltet Ports ab, vermeidet so u.a. Broadcast-Lawinen
Viele Schwächen, ev. lange Umkonfigurations-Zeiten
→ Nachfolger SPB „Shortest Path Bridging“: Erlaubt auch redundante Pfade

- VLAN: Mehrere unabhängige logische Netze auf einem gemeinsamen phys. Netz
  → Unabhängige Adressen, unabhängige Broadcasts (Eingrenzung von Broadcasts!)
  → Sicherheits-Aspekte: Verbindung zwischen VLAN's nur über Router und Firewalls
  → Reduzierung / Einsparung getrennter Kabel und Switches
  → Priorisierung der Pakete (das VLAN-Tag hat auch ein Prioritäts-Feld)
  Entweder Port-basiert (jeder Switch-Port ist einem VLAN zugeordnet)
  oder Tagged (jedes Paket hat ein VLAN-Tag nach dem Layer-2-Header): IEEE 802.1Q

- Optional: Authentifizierung zur Verbindung zu einem WLAN und/oder
  zu einem LAN-Port und/oder zur Aufnahme in ein VLAN nach IEEE 802.1X
  (analog / ident zu WLAN-Auth, z.B. gegen Radius- oder LDAP-Server): EAP-Protokoll

- TCP/IP-Protokoll-Stack, IPv6 (IPv8 ?)
  IP, TCP, UDP (bekannt!?!)
  + ICMP (Layer 3) (2 Zwecke: Ping & Fehlermeldungen)

  Neu auf Layer 4: SCTP („Stream Control Transmission Protocol"):
  Moderneres TCP (verbindungsorientiert, Daten- und Reihenfolge-erhaltend):
  neu: mehrere Datenströme pro Verbindung, die einander nicht blockieren
  kein Syn-Flood-Problem

  Neu auf Layer 4: QUIC („Quick UDP Internet Connections"):
  von Google, Grundlage von HTTP/3 und DNS over QUIC
  breite Open-Source-Unterstützung, heute können alle Browser QUIC
  setzt auf UDP auf, ist aber verbindungsorientiert
  ebenfalls non-blocking multi-Stream
  sollte immer mindestens mit TLS 1.3 verschlüsselt sein
  von Microsoft u.a. für SMB genutzt (wegen Verschlüsselung)
  Ziel: Schneller als TCP, weniger Latenz, bessere Fluss- und Stau-Kontrolle

- TCP/IP:
  Fluss-Steuerung & Sliding Window, Slow Start & Congestion Control:
  Verschiedene Probleme → den einen optimalen Algorithmus gibt es nicht!
  → Verschiedene TCP-Implementierungen nutzen verschiedene Algorithmen
  Ist die Verbindung voll, langsam aber stabil, oder qualitativ schlecht?
  Ist Fairness gefordert? Viele oder wenige Daten pro Verbindung? Lokal / Internet?
  Bufferbloat-Problem?
  ECN-Bit im IP- bzw. TCP-Header („Explicit Congestion Notification"):
  Noch nicht überall einheitlich implementiert
  ICMP „Source Quench" (vom Router oder Ziel-Host): Sinngemäß „Brems dich ein"
  (alt, inzwischen deprecated, potentieller DOS, vom Internet herein oft geblockt)

  TCP Keepalive: Optional (Option SO_KEEPALIVE):
  Ab 2 Std. idle: Keepalive-Pakete senden → Erkennen, wenn der andere weg ist.
  Andere TCP Socket Options: Einige!

- TCP_NODELAY: „Sende sofort, warte nicht auf mehr Daten“
- TCP_CORK (nur Linux): Das Gegenteil: „Warte noch ein bisschen mit dem Senden“

MTU (1500 Bytes pro Paket für IP auf Ethernet):
Heute: Automatische MTU Discovery der gesamten Strecke
Früher: Oft mittendrin nur 512 oder 576 Bytes (z.B. ATM)
→ ICMP-Rückmeldung der kleineren MTU an den Sender, Paket gedroppt
(ICMP oft gefiltert / ignoriert oder gar keine Rückmeldung gesendet)
→ oder automatische Fragmentierung im Router vor der Engstelle
(wenn DF-Bit im Header gesetzt: Keine Fragmentierung, Drop & ICMP-Fehler)
Test z.B. mit ping -f -l in Windows, ping -M in Linux

Fehlermeldungen:
Timeout, Refused, Aborted & Reset, No Route (2x), Down (2x), Unknown Host
TTL („Time to Live“): Keine „Zeit“, sondern Anzahl Hops
Wenn 0: Fehler-Rückmeldung per ICMP → Mechanismus hinter `traceroute`

- Zentrale TCP-Dienste: DHCP (4 Dinge) /BOOTP, DNS, NTP (Zeit),
SNMP = Management von Netzwerk-Geräten (Router, …), Netz-Druckern, …:
Fehlerüberwachung und -Meldung, Konfiguration und Steuerung, …
→ Gefährlich, nach außen blockieren!
Mehrere Versionen, ursprünglich unverschlüsselt (auch Passwort falls überhaupt)
→ Schlechter Ruf „Security is not my problem“
→ Version 3 ist besser, aber komplex und noch nicht weit verbreitet

plus „Triviale“ Ports (meist Dienste ohne separaten Server, direkt im Kernel):
Echo, Daytime und Time, Discard, Chargen

- Höhere Protokolle:
FTP / TFTP, rsync, HTTP → File-Transfer
SMTP, POP, IMAP → Mail
Telnet & Rsh/Rlogin, SSH → remote Text-Konsole, bei SSH auch remote File-Kopie
NNTP → textbasierte Newsgroups „NetNews“, IRC → textbasierter Chat
LPD → alter Druckdienst, IPP → Internet Printing Protocol & CUPS
Syslog → System-Messages-Protokolldienst
LDAP → Verzeichnis-Dienst, Benutzer-Verwaltung
MQTT, Modbus/TCP, OPC/UA → Steuerungen, Home Automation, Internet of Things
SMB/CIFS, NFS (+ SunRPC, NIS), AFS, Cluster-Filesysteme, iSCSI → File-Sharing

- FTP: Ursprünglich Commandline-Tool, heute GUI (z.B. FileZilla)
oder URL ftp:// (aus Sicherheitsgründen nicht mehr)
oder Einbindung als Filesystem im Dateimanager
FTP unterstützt auch Hochladen, Löschen, Dir anzeigen, anlegen & wechseln,
chmod, Datei umbenennen, …
Jahrzehntelang Standard-Download-Protokoll öffentlicher Fileserver:
User „anonymous“, kein Passwort (höflicherweise: E-Mail-Adresse)

FTP kann theoretisch FXP:
Ein Client steuert den direkten Dateiaustausch zwischen 2 FTP-Servern
(*ohne* Umweg über den Client)
Problem FTP: Unverschlüsselt, auch Passwort im Klartext
→ Besser SCP/SFTP (SSH-Protokoll) oder HTTPS mit Put und Get
Problem FTP:
FTP verwendet *zwei* Ports/Verbindungen:
Befehle und Daten getrennt, server-seitig 20 & 21, client-seitig meist >1023
FTP Data war ursprünglich ein Connect vom Server zum Client
→ Problem bei NAT und Firewalls!
→ „Passives" FTP:
Server sendet exklusive Data-Portnummer (meist > 1023) an Client
Beide Connects vom Client zum Server

- Unterschied FTP/TFTP:
TFTP ist UDP (z.B. beim Booten: TCP zu komplex), FTP ist TCP
TFTP ist viel einfacher / kann weniger, ohne Authentifizierung
TFTP Verwendung:
Download von Firmware, BIOS-Updates, Bootloadern beim Booten (BootP / PXE)
ev. gesamtes Betriebssystem bei diskless Thin Clients & diskless Linux
(nach dem Betriebssystem: Root-Filesystem meist mittels NFS, nicht TFTP)
Außer Commandline-Tool TFTP: BIOS, UEFI, uBoot, … können optional TFTP
Malware-Verbreitung! → Von/nach außen blockieren!
rsync: Synchronisation von Verzeichnisbäumen lokal und remote
→ nur die geänderten Dateien werden übertragen!

- SMTP: Sender → Server & Server → Server
POP, IMAP: Server → Empfänger
bei POP: Transfer & Speicherung der Mails am Client (Server löscht nach Abruf)
bei IMAP: Verwaltung & Speicherung der Mails und Mail-Ordner am Server

- SMTP, HTTP, FTP, POP, … sind in der unverschlüsselten Variante alle „Telnet-fähig" (Tool „netcat"): Zeilenorientierte, klartextbasierte, „menschlich lesbare" Protokolle

  Problem Telnet, RSH (und unverschlüsseltes SMTP, POP, IMAP):
  → Passwörter im Klartext!

- Modbus/TCP: Einfachstes standardisiertes IoT-Protokoll
Modbus war ursprünglich für serielle Leitungen (sehr alt), portiert auf TCP:
Ganz einfaches Lesen & Setzen von einzelnen Register-Werten (durchnummeriert)
Optional über TLS

- MQTT: Message Broker bzw. Publisher/Subscriber-Architektur:
Hierarchisches Verzeichnis von Sensor-Werten usw.
Jede Kommunikation läuft über den Broker:
  - Publisher schicken einzelne Werte an den Broker

- Broker sammelt und speichert sie und verteilt sie an alle Subscriber
- Subscriber registrieren sich beim Broker für bestimmte Werte
  und bekommen sie bei jeder Änderung geschickt

Ziel: Endgeräte möglichst einfach, nur Broker aufwändig

- OPC/UA: Von der einzelnen Maschine bis zu SCADA- und ERP-Systemen
  Sehr komplexes Konzept zum Austausch beliebig strukturierter Werte
  (Binärdaten oder XML) im Maschinen- und Anlagen-Steuerungs-Bereich
  (u.a. Siemens, Beckhoff), IEC-Norm, nicht RFC
  Open & Closed Source Implementierungen, Library für viele Programmiersprachen

- SMB („Server Message Block") / CIFS („Common Internet File System"):
  SMB … altes (1983) Filesharing- und Druck-Protokoll
  von IBM für OS/2 & Microsoft für Windows
  ursprünglich NetBIOS/NetBEUI als Transportschicht (nicht IP, nicht routingfähig)
  bald Portierung auf TCP (zuerst NetBIOS über TCP, Ports 137-139,
  dann SMB nativ auf TCP, Port 445)
  verschlüsselt, aber trivial knackbar
  CIFS … „herstellerunabhängiger" Name für SMBv1
  (aus Marketing-Zwecken und rechtlichen Gründen, wegen Normungs-Versuch)
  Zwischenstufe SMBv2: Radikal neu, nicht aufwärtskompatibel
  Heute: SMBv3 (nicht mehr CIFS, nicht mehr NetBIOS, besser verschlüsselt,
  TCP oder QUIC)
  SMB war und ist Verbindungs- und Session-orientiert, nicht stateless:
  File öffnen, stückweise lesen & schreiben
  Authentifizierung & Rechte-Verwaltung über Active Directory
  Open-Source-Implementierung für Linux usw.: Samba

- NFS („Network File System"):
  Klassisches Unix Filesharing, älter als SMB
  Erfunden von Sun Microsystems, immer schon sehr offen & standardisiert (RFCs),
  auf fast allen Unix-Plattformen (& Linux, …) verfügbar, Windows- und Mac-Ports
  v2: Implementierung basiert auf SunRPC (Remote Procedure Calls),
  braucht Portmapper für dynamische Port-Vergabe (→ schlecht bei Firewalls)
  war stateless (!) (d.h. verschickte einzelne 8K-Blöcke auf unabhängige Requests)
  und basierte auf UDP (!) (keine fixen Verbindungen)
  → Server konnte im Betrieb ohne Client-Fehler rebootet werden!
  keine Sicherheit (nur für lokal gedacht)
  zentrale Benutzer-Verwaltung über NIS („Network Information Service"
  = YP Yellow Pages), das frühe Unix-Gegenstück zum Active Directory
  v3: Optional Migration auf TCP (bei größeren Blöcken Geschwindigkeits-Vorteile)
  optional ein bisschen Sicherheit
  v4: TCP, stateful, schneller, verpflichtend verschlüsselt & sehr sicher (u.a. Kerberos)
  nur mehr 1 fixer Port nötig, Firewall-freundlich

parallele Zugriffe, Multi-Pathing
nicht mehr unter Kontrolle von Sun, sondern von IETF geleitet

- AFS = „Andrew File System“, heute meist in der Version von OpenAFS
  Universitäts-Entwicklung (CMU), standardisiert, inzwischen Open Source
  immer schon sicher (Kerberos)
  für Größt-Systeme (tausende / zehntausende Clients) → selten eingesetzt
  „Distributed“ Filesystem, d.h. eine logische Sicht auf viele verteilte Server
  → Vorstufe von Cluster-Filesystemen
  Intensives Caching auf lokaler Maschine und Replikationsservern
  → Geöffnete Files werden als Ganzes auf den Client transferiert
  und erst nach dem Schließen zurück auf einen Server geschrieben
  → Eingeschränktes Offline-Arbeiten möglich
  → Schlecht geeignet für Datenbanken und Files mit mehreren änderndern Clients,
  kein striktes Locking, keine laufende Synchronisation
  Einige Nachfolger-Projekte (vor allem „Coda“)
- Cluster-Filesysteme: Konzeptioneller Unterschied:
  - Netzwerk-Filesystem:
    Ein Server verwaltet exklusiv ein lokales Filesystem
    auf (zumindest logisch) lokalen Platten exklusiv für ihn
    → die Netzwerk-Ebene liegt logisch nur zwischen Clients und Server
    → Der Server ist Single Point of Failure und limitiert die Performance
  - Cluster-Filesystem:
    Mehrere Server haben über ein Netzwerk **gleichzeitig** Zugriff auf die Platten
    (auf Block-Ebene wie bei lokalen Platten-Anschlüssen, nicht auf File-Ebene)
    und verwalten auf diesen Platten **gemeinsam und gleichzeitig ein Filesystem**
    → Zwei Netzwerk-Ebenen: Client ↔ Server und Server ↔ Plattenlaufwerke
    → Die Server sind gleichberechtigt, kein fixer Master
    daher auch kein Single Point of Failure, hohe Redundanz
    → Jeder Client kann über jeden Server zugreifen,
    daher Skalierbarkeit und hoher Durchsatz

  Aber: Die Logik von Cluster-Filesystemen ist **sehr komplex**
  (Synchronisation der Zugriffe & Locking, gemeinsame Platzverwaltung, …),
  denn (Meta-) Daten-Änderungen sind konzeptionell auf allen Servern synchron
  → Cluster-Filesysteme sind auf gleicher Hardware **langsamer** als normale Fileserver

  Anwendung: Hochverfügbarkeit, Supercomputing, Systeme mit vielen Clients
  Beispiele:
  - Lustre (häufig bei Supercomputern & großen Rechenzentren)
  - Ceph (Open Source + Windows, hochverfügbar,
  verteilte File-, Objekt- und Block-Storage, Block vor allem für VM-Laufwerke)
  - GlusterFS (Red Hat, viele Plattformen)

- OCFS2 (von Oracle für Linux, vor allem für Oracle Datenbanken)
- … und viele proprietäre, z.B. Veritas Cluster FS

- Unterhalb von Filesystemen:
  Netzwerk-Protokolle zum Zugriff auf Platten auf Block-Ebene (nicht File-Ebene):
  - FC („Fibre Channel“, british spelling!):
    Eigene Hardware (oder FCoE = Ethernet)
    Eigener Software-Protokoll-Stack aufbauend auf SCSI
    Für SAN's = Storage Area Networks (einige km)
    Nicht TCP-kompatibel, nicht Internet-fähig, teuer
    (aber man kann FC über IP und IP über FC tunneln)
    Historisch in großen Rechenzentren etabliert
  - iSCSI („Internet SCSI“)
    Das seit Jahrzehnten übliche SCSI-Zugriffsprotokoll auf Platten
    (und Bänder, Band-Roboter, optische Laufwerke):
    – Original SCSI: Parallele Kupferkabel zur Platte
    – SAS („Serially attached SCSI“): Physisch seriell (ident zu SATA), logisch SCSI
    – FC: FC ist auf Software-Ebene ein SCSI-Dialekt über FC-Hardware
    – iSCSI aufgesetzt auf TCP über beliebige Netze (häufig, häufiger als FC)
    – HyperSCSI: SCSI direkt (ohne IP) auf Ethernet-Frames (sehr selten)

- AS-Nummern, globales Routing (BGP), globale Netz-Struktur, Peering, …
  Unterschied IGP-Protokolle (OSPF, RIP) / EGP-Protokolle (BGP)
- NAT, SOCKS, DMZ, Paketfilter, Firewalls
- Verschlüsselungen (SSL/TLS, VPN, IPsec…)???

Offen:

- Multicast, IGMP???
  Zeroconf / Bonjour / Avahi / mDNS???
- Point-To-Point-Protokolle (SLIP, PPP / PPPoE, PPTP, L2TP, …)???
- Radius???
- VoIP, SIP, ENUM, RTP???
- WLAN-Protokolle???
- Netzwerk-Config & Admin unter Linux???
- Netzwerk-Analyse-Tools (Wireshark, nmap, netcat, traceroute…)???
- Socket-Programmierung in C???