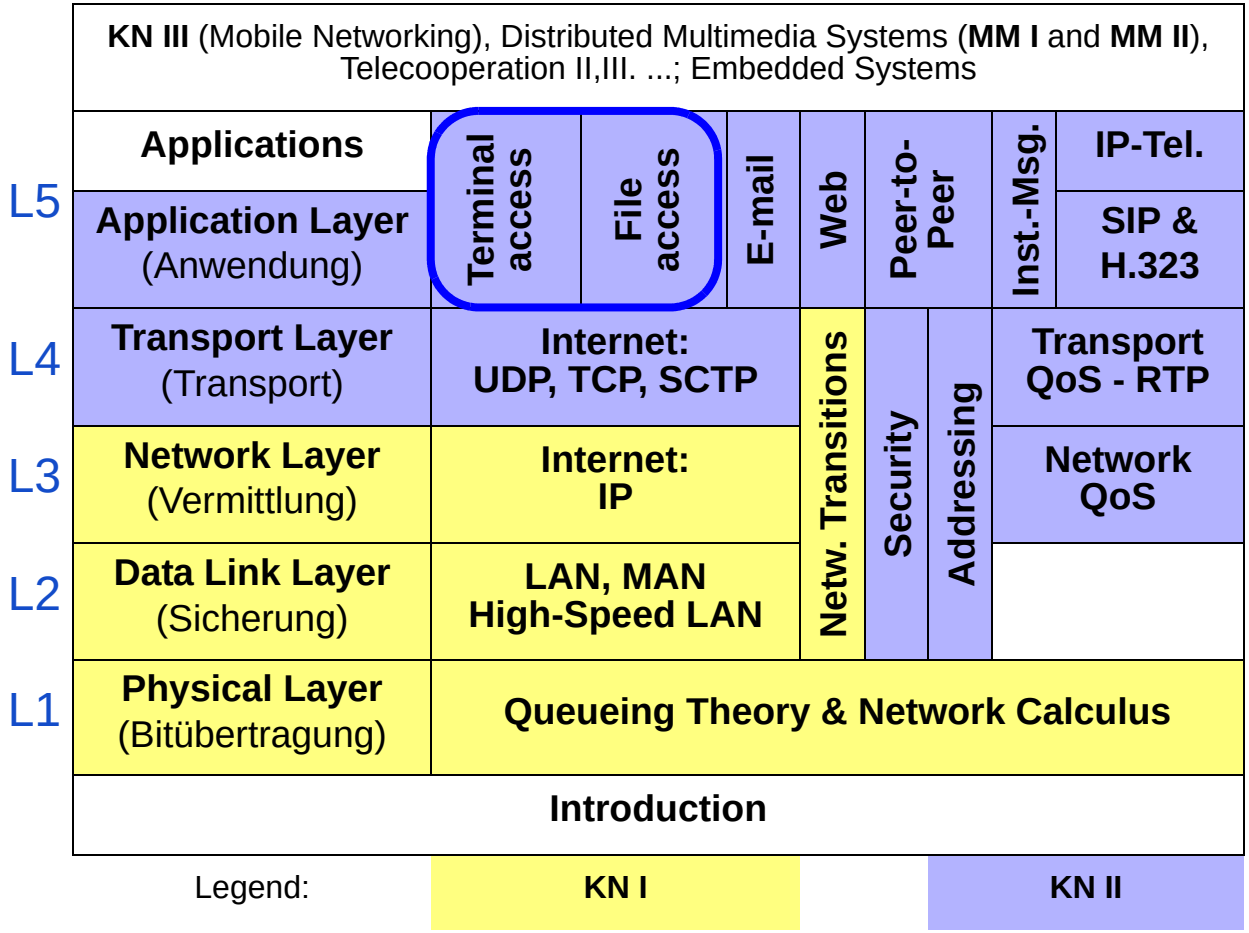
Communication Networks II
Network Applications - Online Services

Prof. Dr.-Ing. **Ralf Steinmetz**

TU Darmstadt - Technische Universität Darmstadt,
Dept. of Electrical Engineering and Information Technology, Dept. of Computer Science
KOM - Multimedia Communications Lab
Merckstr. 25, D-64283 Darmstadt, Germany, Ralf.Steinmetz@KOM.tu-darmstadt.de
Tel.+49 6151 166151, Fax. +49 6151 166152

httc - Hessian Telemedia Technology Competence-Center e.V
Merckstr. 25, D-64283 Darmstadt, Ralf.Steinmetz@httc.de

Scope


KN III (Mobile Networking), Distributed Multimedia Systems (MM I and MM II), Telecooperation II,III. ...; Embedded Systems
Applications
Application Layer (Anwendung)
L5
Terminal access
File access
E-mail
Web
Peer-to-Peer
Inst.-Msg.
IP-Tel.
SIP & H.323
Transport Layer (Transport)
L4
Internet: UDP, TCP, SCTP
Netw. Transitions
Security
Addressing
Transport QoS - RTP
Network Layer (Vermittlung)
L3
Internet: IP
Network QoS
Data Link Layer (Sicherung)
L2
LAN, MAN High-Speed LAN
Physical Layer (Bitübertragung)
L1
Queueing Theory & Network Calculus
Introduction
Legend:
KN I
KN II

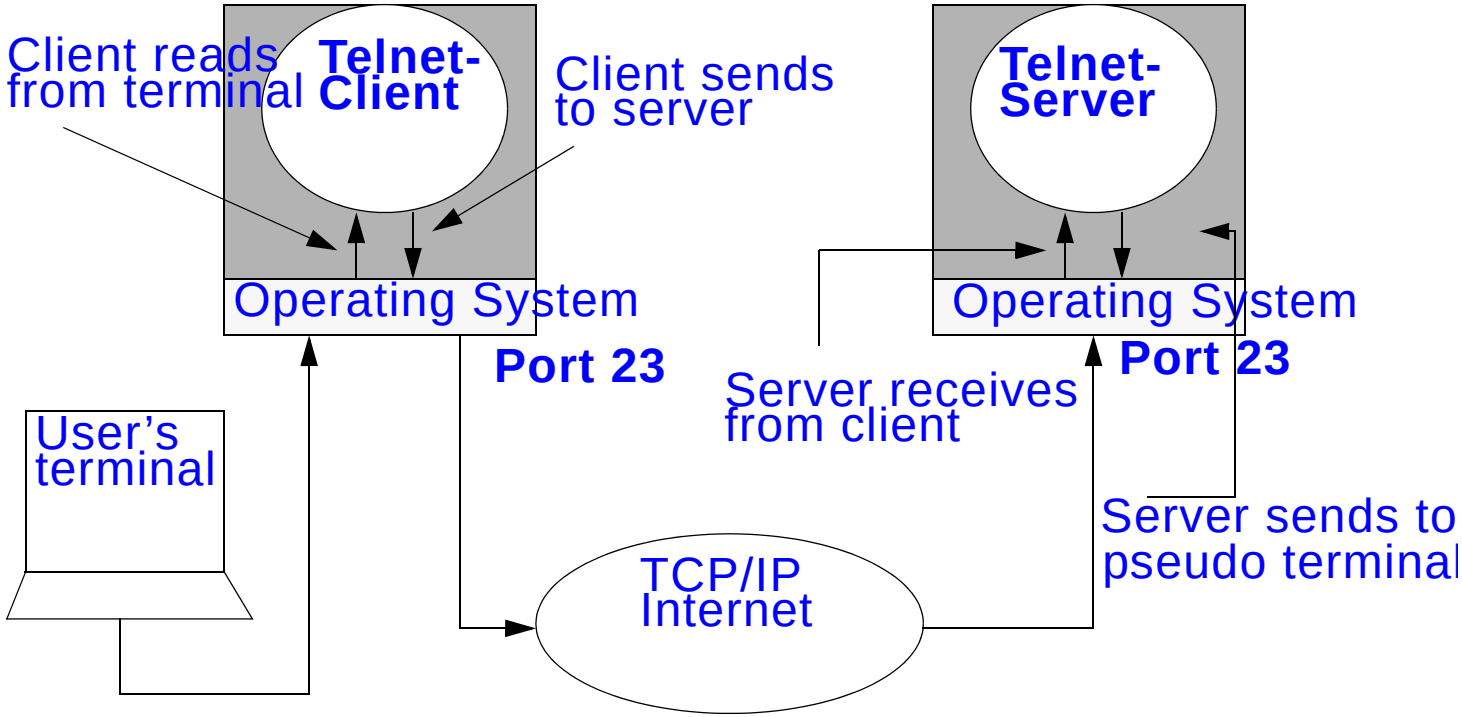
Overview

1. Remote Login: Telnet

2. Data Transfer - File Transfer Protocol (ftp)
   - 2.1 Example of an ftp Session (User's Perspective)
   - 2.2 Example for ftp Commands (System's Perspective)
   - 2.3 Additional Information

3. Network File Systems: nfs, afs/dfs, and further
   - 3.1 Network File System (nfs)
   - 3.2 Andrew File System (AFS or DFS)
   - 3.3 Further Systems for Networked Access to Files

1. Remote Login: Telnet



Functionality:

- **remote login**
  - "Network Virtual Terminal"
- **full screen, i.e. scrolling but no graphics capability**
- **simple terminal protocol**
- **permits negotiations of options**
  - e.g. data transfer: binary or ASCII

Remote Login: Telnet (2)

Implementation

- based on TCP connection between client and server
- uses Port 23
- RFC 854: Telnet protocol specification
  J. Postel, J.K. Reynolds. May-01-1983 and supplements

Telnet: example

```
[steinmetz on dumbek] ~ $ telnet flute
   Trying 130.83.139.139...
   Connected to flute.kom.tu-darmstadt.de.
   Escape character is '^]'.
   Technische Universitaet Darmstadt
   Multimedia Kommunikation
login:
   login: steinmetz
   Password:
Password: ...tipp..tipp..tipp..
   [steinmetz on flute] ~ $
```

But Telnet is insecure:

- clear text password

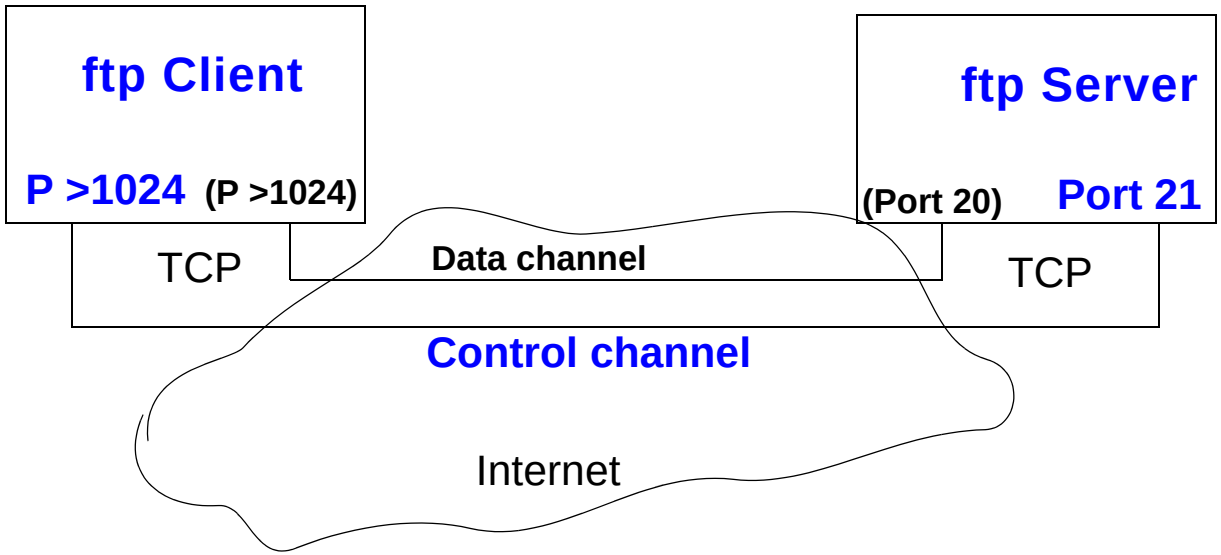
  ⇒ Hence, nowadays often switched off and replaced by ssh

2. Data Transfer - File Transfer Protocol (ftp)

File Transfer allows for

- **file transfer intiation**
  - send (put, mput)
  - receive (get, mget)
- **file transfering**
  - binary
  - textual mode (ascii)
- **file manipulation**
  - delete (del)
  - rename
- **directory operations**
  - print working directory (pwd),
  - list directory's contents (ls, dir)
  - create /remove directories (mkdir, rmdir)
  - change directory (cd)
- **user identification or "anonymous ftp"**
  - of an account/name (user)
  - identification (password)
- **additional possibilites (help, etc.)**

File Transfer - File Transfer Protocol (2)



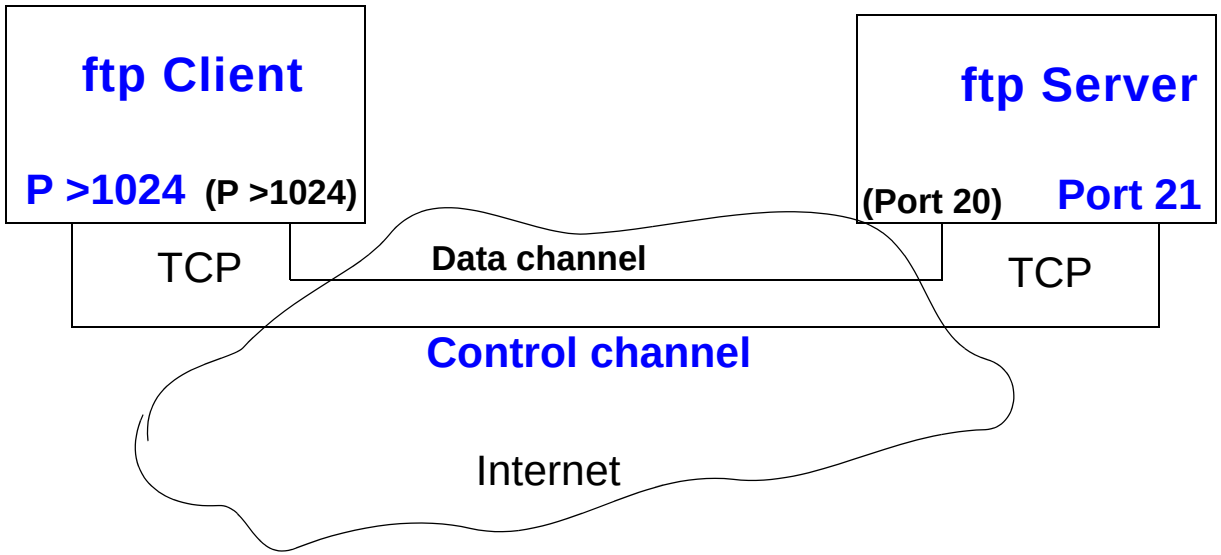
Functionality

- **uses TCP for data communication**
- **ftp client runs as a programm within the user's address space**

Some Remarks

- **no integration into local file system**
  - i.e. no transparency
- **does not use a spooler**

Data Transfer - File Transfer Protocol (ftp) (3)



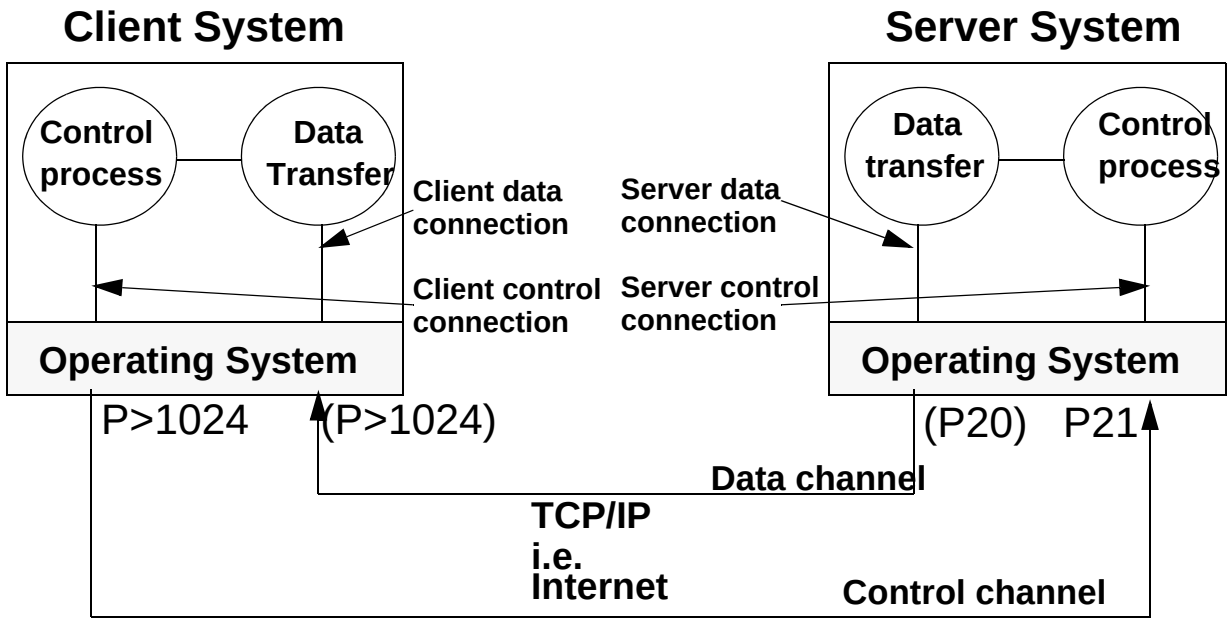
Commands

- **transmitted as a 4-character sequence plus options**
- **e.g. PASS xyz**

Response

- **sequence consisting of 3 numbers**
- **first number indicates error status**
  - 1,2,3: no error
  - 4,5: error

Data Transfer - File Transfer Protocol (ftp) (4)



TCP control connection

- **exists while the systems interact**
- **therefore can also execute other functions during data transfer**
  - because of 2 connections

TCP data channel

- **for data transfer and data of directory listings (multiline response)**
- **reconnects and disconnects for each data transfer**
- **connection set-up is done in reverse direction**

2.1 Example of an ftp Session (User's Perspective)

```
$ftp
   ftp>
ftp> open ftp.kom.tu-darmstadt.de
   Connected to conga.kom.tu-darmstadt.de.
   220 conga.kom.e-technik.tu-darmstadt.de FTP server
   (Version wu-2.6.1-16) ready.
   User (conga.kom.tu-darmstadt.de:(none)):

User (conga.kom.tu-darmstadt.de:(none)): anonymous
   331 Guest login ok, send your complete e-mail
   address as password.
   Password:

Password: **my-passwort e-mail-Adr not-displayed**
   230-**********************************
   230-Welcome to KOM's FTP-Server!
   230 Guest login ok, access restrictions apply.
   ftp>
```

Example of an ftp Session (User's Perspective) (2)

```
ftp> ls
   200 PORT command successful.
   150 Opening ASCII mode data connection for file
   list.
   pub
   priv
   incoming

   226 Transfer complete.
   21 bytes received in 0.017 seconds (1.2 Kbytes/s)
   ftp>

ftp> pwd
   257 "/" is current directory.
   ftp>
```

Example of an ftp Session (User's Perspective) (3)

```
ftp> get
   (remote-file)
(remote-file) pub/index.html
   (local-file)
(local-file) i.tmp
   200 PORT command successful.
   150 Opening ASCII mode data connection for pub/
   index.html (1339 bytes).
   226 Transfer complete.
   local: i.tmp remote: pub/index.html
   1375 bytes received in 1.6 seconds (0.86 Kbytes/s)
   ftp>
ftp> close
   221 Goodbye.
   ftp>
ftp> quit
   $
```

2.2 Example for ftp Commands (System's Perspective)

Here: telnet has been used to emulate ftp

```
$ telnet conga 21
   Trying 130.83.139.247...
   Connected to conga.kom.tu-darmstadt.de.
   Escape character is '^]'.
   220 conga.kom.tu-darmstadt.de FTP server (Version
   wu-2.4.2-academ[BETA-12](1) Wed Mar 5 12:37:21 EST
   1997) ready.

USER ftp-guru
   331 Password required for ftp-guru.

PASS 4to1a-kom
   230 User ftp-guru logged in.
PWD
   257 "/home/ftp-guru" is current directory.
```

Example for ftp Commands (System's Perspective) (2)

```
HELP
   214-The following commands are recognized (* =>'s
   unimplemented).
     USER    PORT    STOR    MSAM*   RNTO    NLST    MKD     CDUP
     PASS    PASV    APPE    MRSQ*   ABOR    SITE    XMKD    XCUP
     ACCT*   TYPE    MLFL*   MRCP*   DELE    SYST    RMD     STOU
     SMNT*   STRU    MAIL*   ALLO    CWD     STAT    XRMD    SIZE
     REIN*   MODE    MSND*   REST    XCWD    HELP    PWD     MDTM
     QUIT    RETR    MSOM*   RNFR    LIST    NOOP    XPWD
   214 Direct comments to ftpadmin@kom.tu-darmstadt.de.
```

.. and so on

```
QUIT
   221 Goodbye.
   Connection closed by foreign host.
```

2.3 Additional Information

History

- **First specification**
  - 1971 form M.I.T.
  - RFC 114
- **Variations**
  - 1971 - 1985
  - more than 10 additonal changes and enhancements
- **Present version**
  - by J. Postel (and J.Reynolds)
  - as of Oct. 1985
  - RFC 959

Further details by experiments

- **as telnet session (see above)**
- **with sniffer**
  - e.g. make use of

    `www.ethereal.com`
    `www.packetyzer.com`
  - and record a simple ftp session

Simple File Transfer: Trivial File Transfer Protocol (TFTP)

Based on the UDP transport protocol

- **simpler**
- **less complex to implement, and less code**

Pure file transfer service

- **e.g. no possibility to view file system on remote system**
- **e.g. no possibility of authentication**

Application

- **e.g. bootstrapping over the network**

3. Network File Systems: nfs, afs/dfs, and further

File Transfer Protokoll

- **explicit data request**
- **explicit commands**

Integration into the file system

- **implicit data transfer**
- **benefit: transparency**
  - locally and remotely stored data
    - treated the same/similarly
  - all programs use the data by means of read/write accesses

3.1 Network File System (nfs)

Network file system (nfs) for remote access to files in the network

- **i.e. access to parts of files (as opposed to ftp)**
- **transparent access to files in remote file systems**

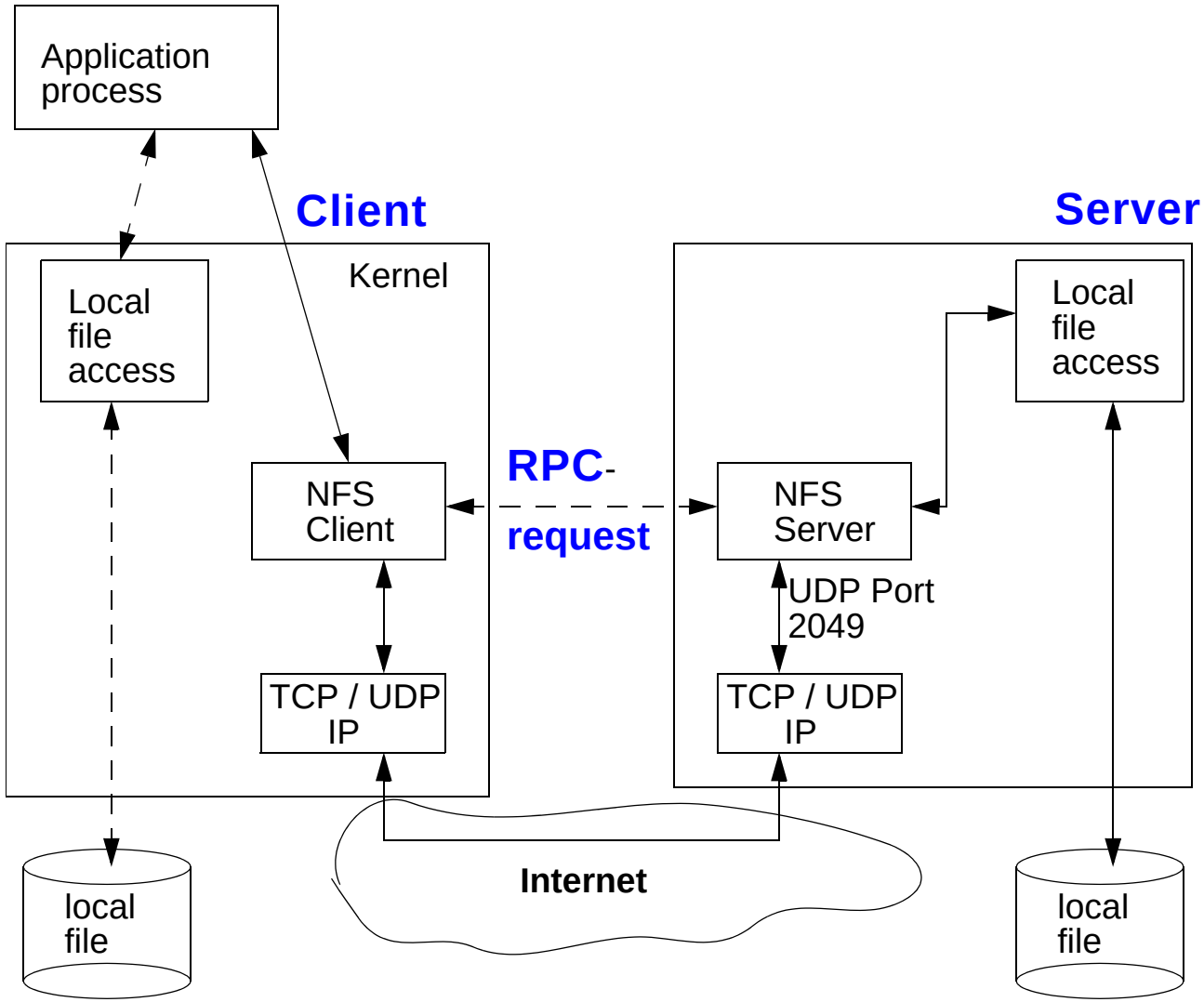
History

1984	**announcement**
1985	**first product presented by SUN**
1986	**porting for system V - release 2**
1986	**NFS 3.0: improved yellow pages (localization of files) and PC-NFS**
1987	**NFS 3.2: file locking**
1989	**NFS 4.0: encoding**
1989	**licensed by 260 suppliers**
....	

And as IETF rfc

1989 rfc 1094	**NFS: Network File System Protocol Specification**
....	
1999 rfc 2624	**NFS Version 4 Design Considerations**
2000 rfc 3010	**NFS version 4 Protocol**
....	

nfs Architecture



nfs Characteristics

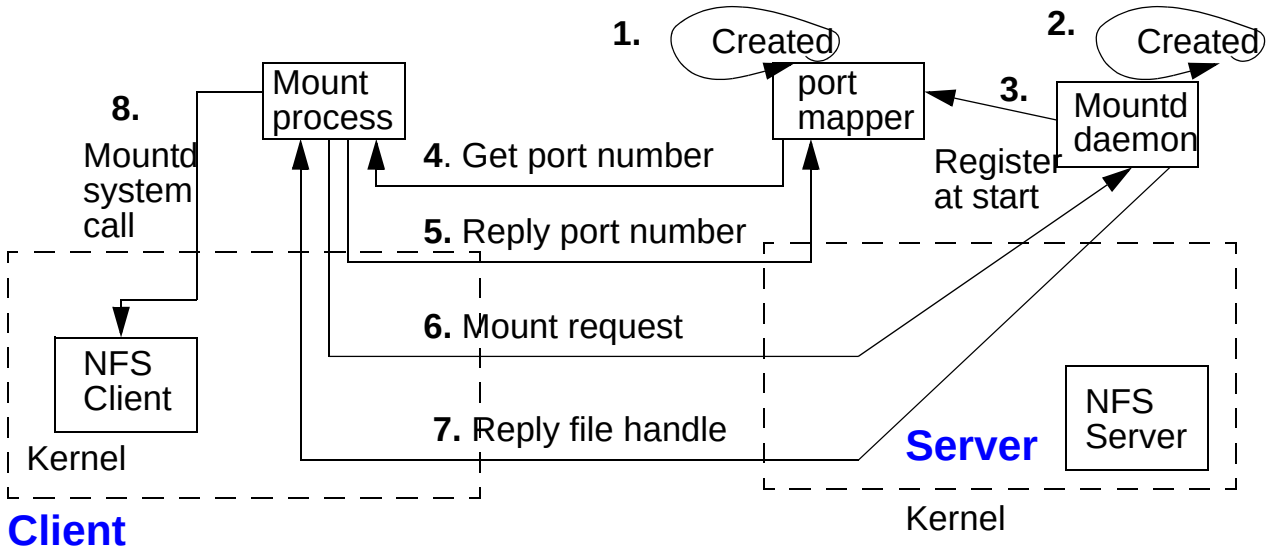
in general

- **client-server model**
- **communication**
  - originally only UDP
  - TCP enables nfs over WANs
- **no presentation services, only byte stream read and write**
- **nodes import/export directories**
- **integrated into the operating system/file system**

widespread because

- **suitable for heterogeneous networks**
- **inexpensive**
- **open system (public specification)**
- **availability**
  - simple to port to new platforms
  - public reference implementation
  - by now standard on almost all UNIX systems
    - component of e.g. UNIX-V-Release-4 since 1989
    - and e.g. PC-NFS for PCs

Implementing nfs: Mount Protocol



NFS

- **protocol for file access only (read and write)**
- **providing remote file access by MOUNT protocol**

MOUNT

- **connects remote file system with local directory**
- **whole remote file tree is mounted into local directory**
- **followed by NFS remote file access just as if accessing local data**

Implementing nfs: Mount Protocol (2)

MOUNT and NFS are

- **separate protocols**
- **MOUNT or mountd supply information for NFS or nfsd (e.g. system name and paths)**

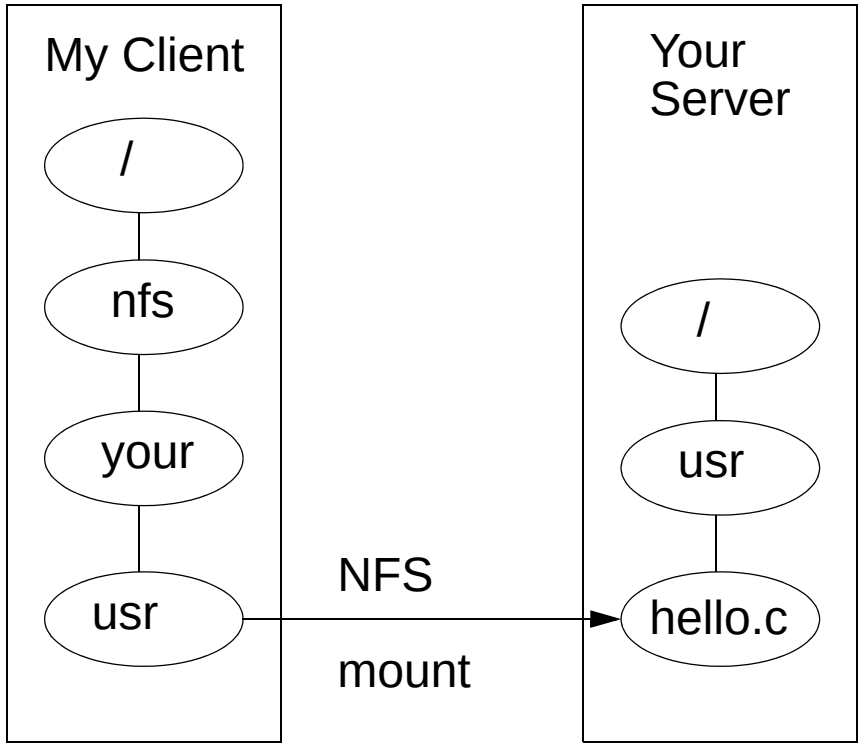
Mountd and nfsd

- **daemons with regard to Unix**
- **automatically started when the server is booted**
- **nfsd activates the NFS server code in the operating system**

Mount process

- **at nfs server**
  - a “file handle” (unique file system control block) is generated
- **to nfs client**
  - the file handle is returned
- **nfs client**
  - uses file handles when accessing the remote (sub-)directory tree

Implementing nfs: Mount Protocol (3)



Example:

```
mount -t nfs your:/usr /nfs/your/usr
```

- **i.e. address**
  /nfs/your/usr/hello.c actually is /usr/hello.c

Specific nfs Problems

Security

- **general problem**
  - link to network always represents a potential security problem
  - Ethernet packets can be easily tapped (intercepted)
- **solution**
  - with version 4.0: NFS supports encryption
  - only privileged ports (<1024) permit data access
  - similar mechanism for MOUNT
  - comparison of
    - internet address
    - client name with /etc/hosts

Data consistency

- **problem:**
  - several remote NFS clients have simultaneous write access
- **lock manager**
  - allows for files to be locked
  - service parallel to NFS (lockd)
  - with NFS version 3.2
  - **NOT** part of the operating system (e.g. not in SUN-OS)
  - no deadlock recognition

3.2 Andrew File System (AFS or DFS)

Overview

- **functionality similar to NFS**
- **mutual authentication requested when contact is established**
  - user **TOKENS** authenticate communication mutually
  - tokens generated at login
  - stored in the AFS cache manager
- **security**
  - token always has to be submitted when a service is requested
  - access protection of the AFS file tree
    - through Access Control Lists (ACL's)
  - multi-layered administration privileges

Further development

- **AFS 4.0 is called the DISTRIBUTED FILE SYSTEM (DFS)**
  - file system component in the
    - Distributed Computing Environment (DCE) of the Open Software Foundation (OSF)
  - not to be confused with the Microsoft DFS system

Authentication in AFS through Kerberos

Mutual authentication

- **mutual identity proof (user, AFS authentication server)**
  - user:
    - has to know the password
  - server:
    - has to decrypt and
    - has to respond to a message which the login process has encrypted with the user password

File access:

- user: AFS cache manager on user workstation checks
  - user identity by using the token
- server: AFS file server checks
  - identity by decrypting the token and
  - by responding with the requested service

In general

- **communication partners know "Shared Secret"**
  - shared secret in form of an encryption key
- **in AFS SIMPLE and COMPLEX MUTUAL AUTHENTICATION**
  - depends on the no. of used keys and
  - depends on the no. of participating partners

Simple Mutual Authentication

In general

- usually the first step of authentication during login

Challenge-response process

1. **Login program on AFS client workstation**
   - sends **CHALLENGE MESSAGE**
     - with encryption key encoded by login (computed from user password -afs-password)
   - to authentication server (using AFS database server)
2. **Authentication server**
   - decrypts message with the user password listed in the database
3. **Authentication server**
   - generates response, which also contains original message
4. **Authentication server**
   - sends this response encoded with the same key back to the login process
5. **Login process**
   - decrypts and verifies the response with the original message

Complex Mutual Authentication

1 (step): simple mutual authentication
2: TICKET GRANTER (auth. server comp.) supplies TOKEN to the login process

Token contains

- server ticket Tc
  - confirmation for successful user identity verification
  - Tc is encrypted with server encryption key Ks
    (shared secret of the AFS server and the auth. server of one cell)
  - thus client cannot decrypt Tc
- session key Kc,s
  - random number issued by the ticket granter
  - shared secret between client and AFS servers
  - Kc,s is part of the tickets (encrypted with the server key) and also unencrypted in the token
- flag for which servers the ticket is valid
- ticket validity period
- complete ticket encrypted with encryption key;
  the key is known to the login process and to the auth. server

AFS cache manager

- does not know the user password
  does not know the encryption key derived from it
- is used for storing the encrypted ticket and session keys

File Access in AFS

1. **On behalf of the user, the AFS cache manager sends the encrypted ticket and the requested service (encrypted with the session key) to AFS**
2. **The AFS server deciphers the ticket to learn the session key (session key is the shared secret)**
3. **AFS server sends response encrypted with the session key**

Reliability Concept

- **client can recognize session key only**
  - if it was able to decipher the token
- **deciphering only possible**
  - if the user knows the correct password at login
- **AFS server can learn session key only**
  - by deciphering the ticket
- **ticket granter only generates valid ticket**
  - if the identity of the user has been proven.
  - The ticket is encrypted with the server encryption key
- **only ticket granter and AFS server know the server encryption key**
- **critical:**
  - AFS database server (auth. database, auth. server, ticket granter)

Access Protection in AFS

Access Protection through Access Control Lists (ACLS's)

- **one ACL per directory**
- **defines file access rights**
  - lookup (to display ACL and files)
  - insert (to generate directory entries)
  - delete (to remove directory entries)
  - administer (to change the ACL of a directory)
  - read (to reads the file, if the UNIX-Bit r has been set for the owner, analog execution if x has been set)
  - write (like to read, if w has been set for owner)
  - lock (to set and to remove advisory locks)
- **20 entries per ACL**
  - user and group name with access rights
    - one-character abbreviations for access rights
  - users themselves can
    - define (up to 20) groups (users or systems)
    - place their own or foreign groups into an ACL

3.3 Further Systems for Networked Access to Files

Coda

Coda networked filesystem

- origin in AFS2
- has been developed at CMU since 1987
- by the systems group of M. Satyanarayanan

It has many features that are very desirable for network filesystems.

Features (Coda has several features not found elsewhere)

- Disconnected operation for mobile computing
- Is freely available under a liberal license
- High performance through client side persistent caching
- Server replication
- Security model for authentication, encryption and access control
- Continued operation during partial network failures in server network
- Network bandwith adaptation
- Good scalability
- Well defined semantics of sharing, even in the presence of network failures

`http://www.coda.cs.cmu.edu/`

Samba

features

- File & print services
- Authentication and Authorization
- Name resolution
- Service announcement (browsing)

"Common Internet File System" CIFS

- all features managed through this protocol suite
- name introduced by Microsoft

Samba is

- an open source CIFS implementation,
- available for free from the

```
http://samba.org/ mirror sites.
http://de.samba.org/samba/samba.html
```