UNIVERSITÄT DUISBURG ESSEN

Internet-Technologie & Web Engineering
File Transfer Protocol (FTP)

Dr.-Ing. Matthäus Wander

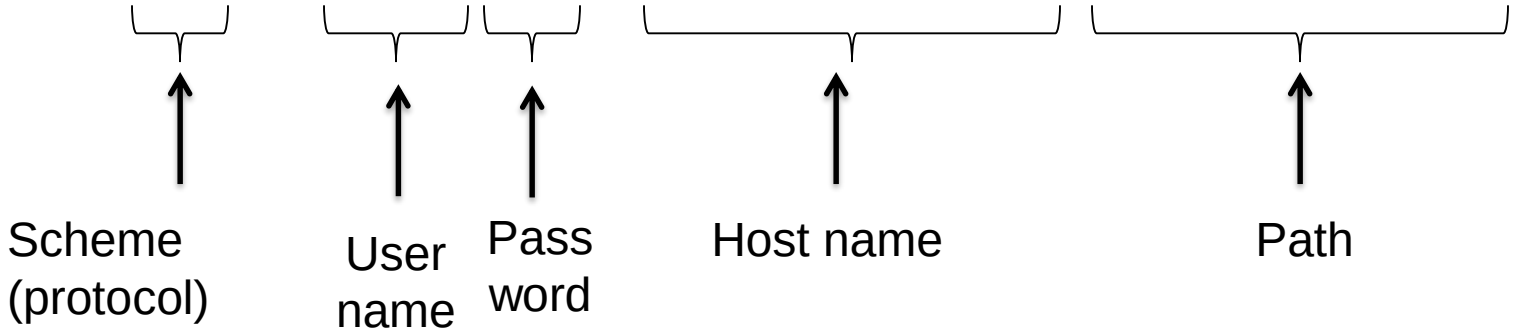
Universität Duisburg-Essen

FTP: Overview

- IETF Internet Standard
  - Specified in RFC 959
- Reliable TCP-based file transfer
- Features: directory handling, access control, resuming partial transfers
- User logins but no encryption
  - Later extension: secure FTP connections with SSL/TLS
- Text-based protocol (but supports binary data)

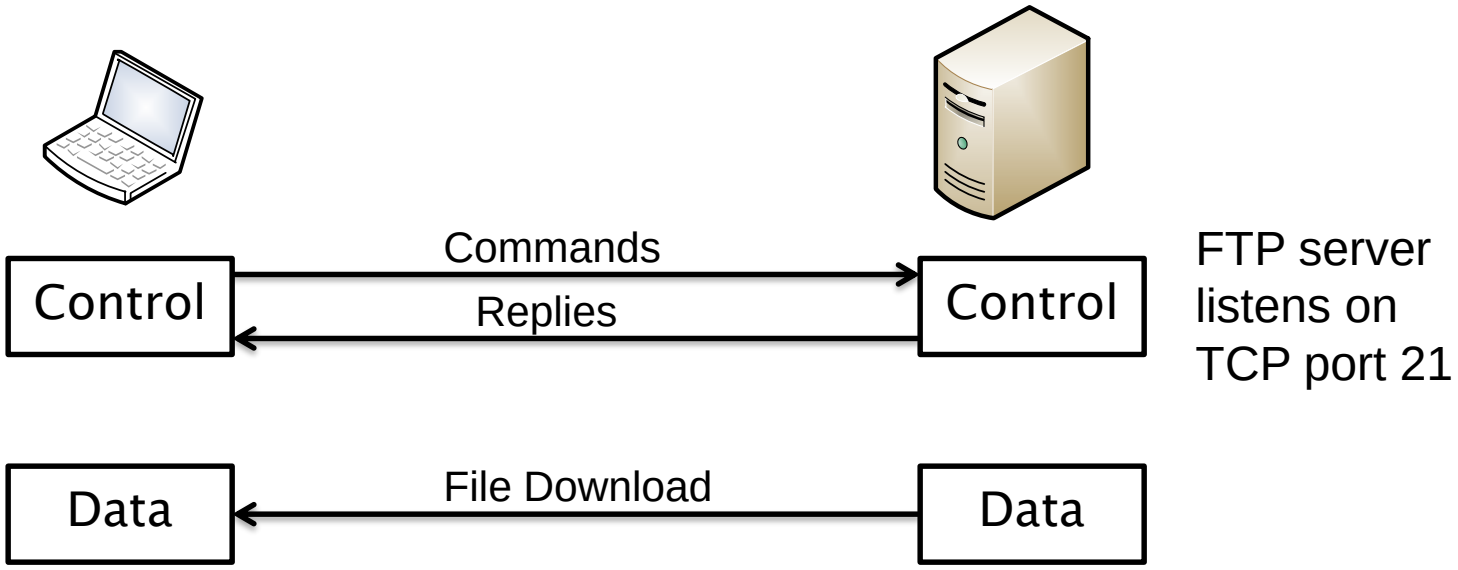
Uniform Resource Locator

- ftp://user:pw@example.net/path/file.txt

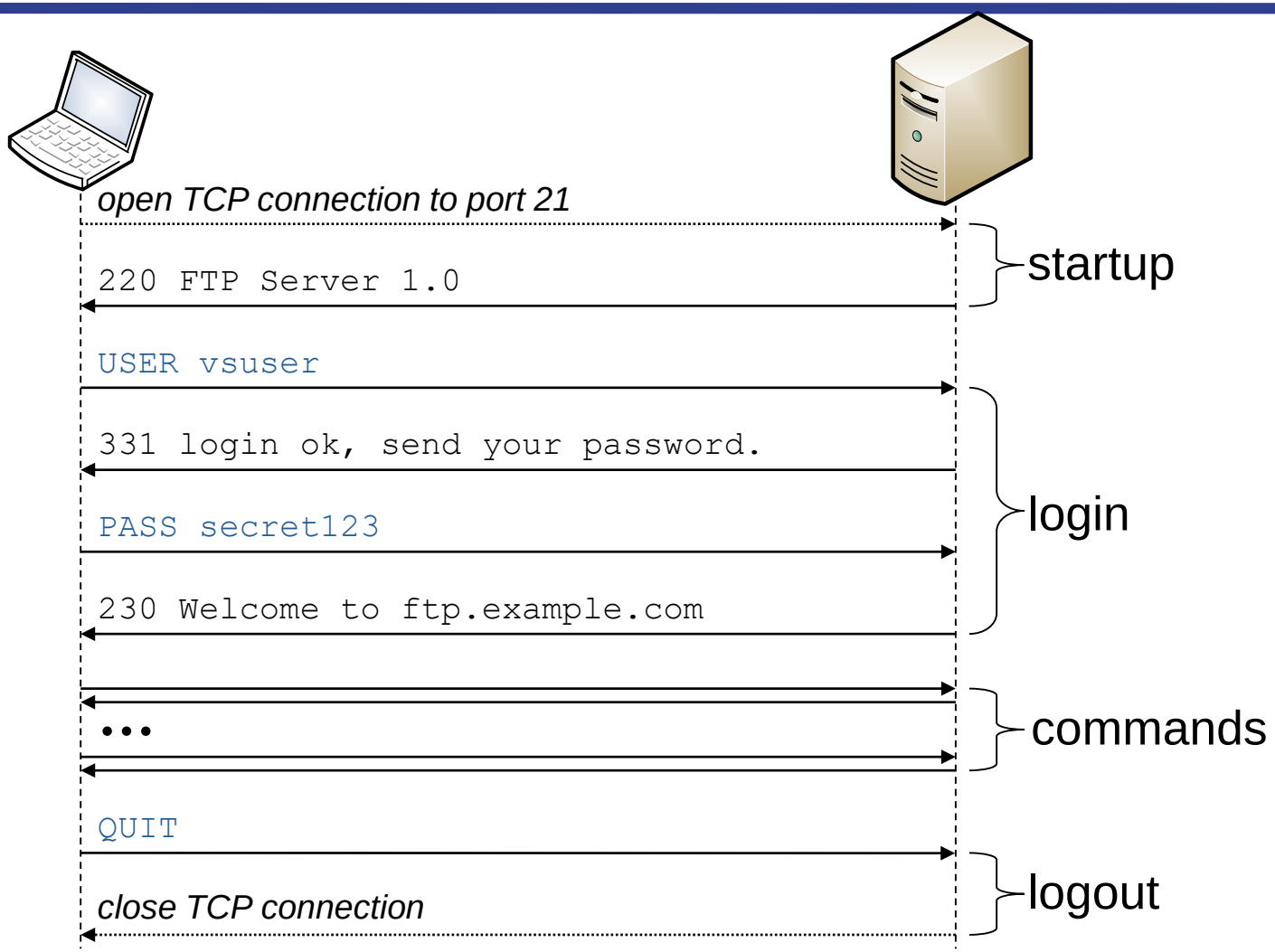


Network Protocol

- Separate TCP connections for control and data



Control Connection



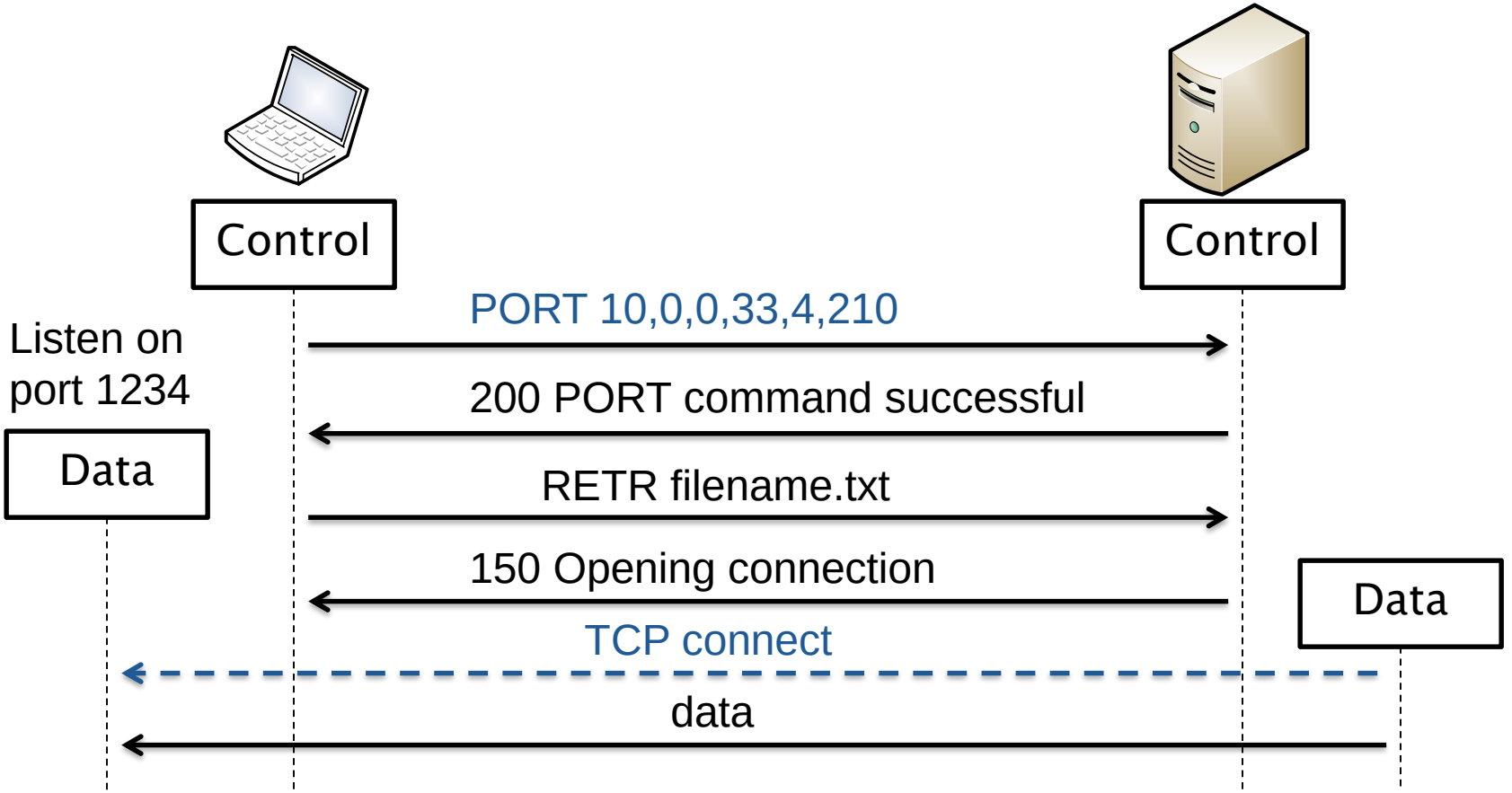
Control Connection (2)

- Test-based control connection
  - ASCII text as specified by TELNET protocol
- Client sends commands, server replies
  - Client: `MKD foo`
  - Server: `257 “foo“ directory created`
- Status code evaluated by client program
  - E.g. 257 = ok; 502 = syntax error
- Status text is auxiliary information for human

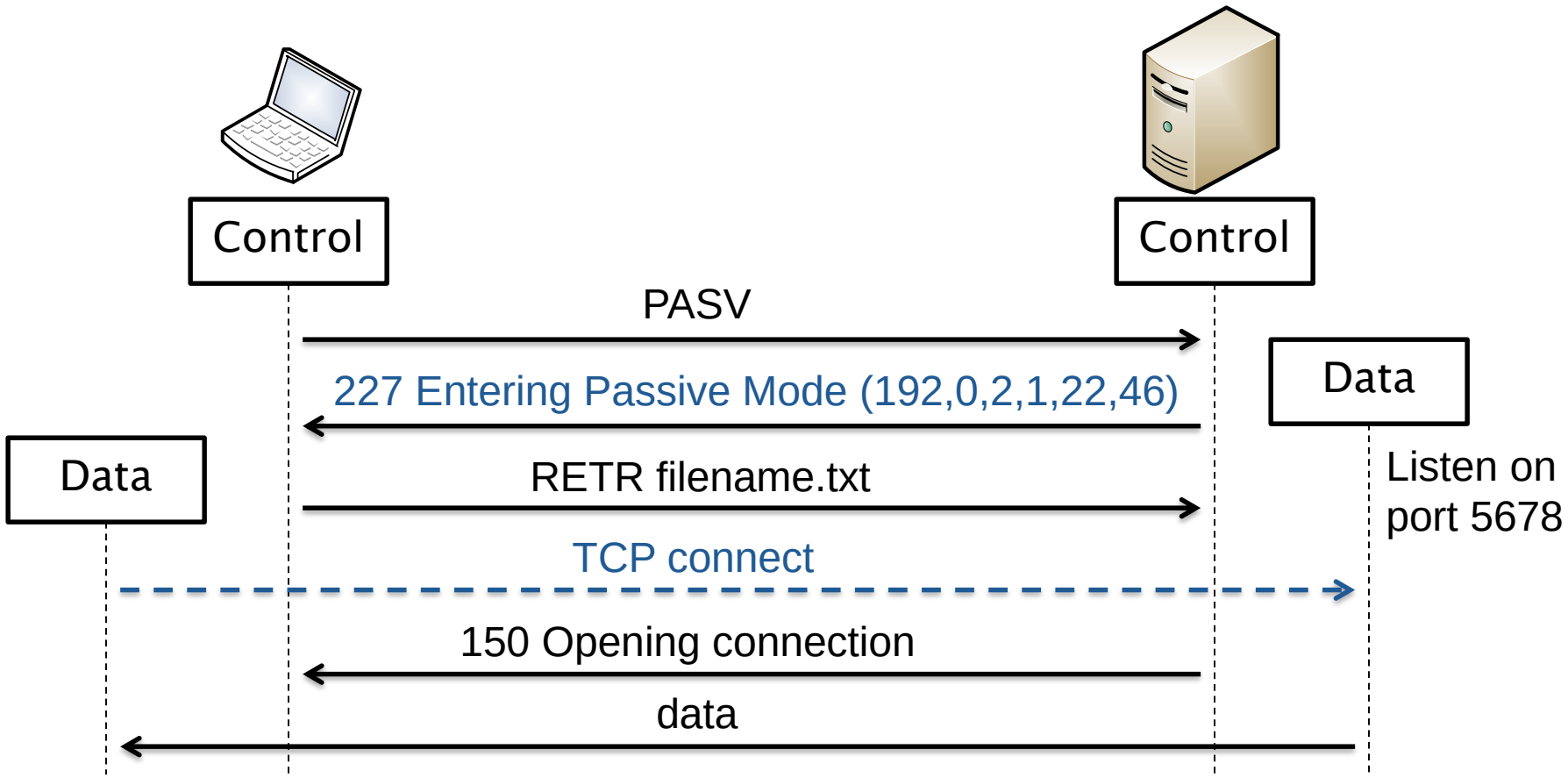
Data Connection

- Used for file transfer and directory listings
- Data types:
  - ASCII
  - 8-bit binary („image“)
- Opened/closed as needed
  - Usually new data connection for each file transfer
- TCP/IP endpoint signaled in control connection
  - Format: `132,252,181,48,212,91`
  - IPv4 address and TCP port

Active Mode


Control
Control
PORT 10,0,0,33,4,210
Listen on
port 1234
200 PORT command successful
Data
RETR filename.txt
150 Opening connection
Data
TCP connect
data

Passive Mode

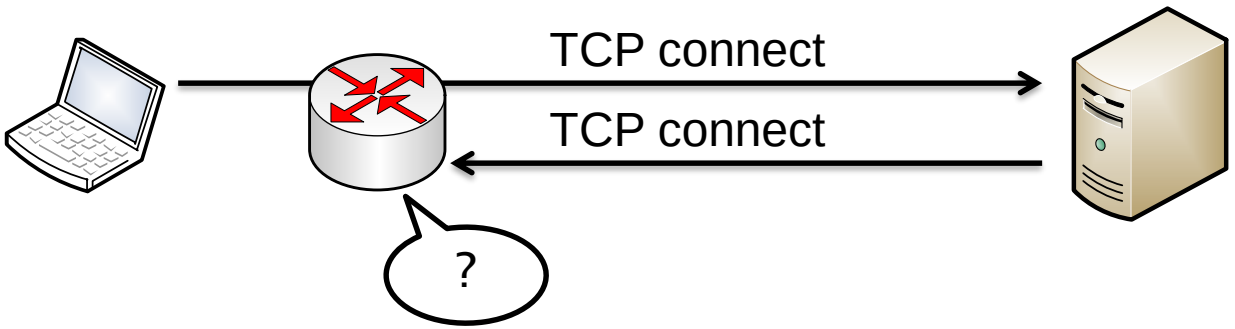


Why Separating Data and Control?

- Myth: allow multiple, simultaneous downloads
  - Fact: not supported by RFC 959
  - "… sending another command before the completion reply would be in violation of protocol; …"
- Design choice dates back to 1971 (RFC310)
  - Bytes were not always 8 bit, TCP did not exist yet
  - Control must use 8-bit bytes (specified by TELNET)
  - Data connection may use another transport method and representation
  - E.g. PDP-10 works on 36-bit bytes

Problems of Data/Control Separation

- Today most clients are masqueraded by NAT
  - Active mode won‘t work, use passive mode



- TCP connection establishment incurs overhead
  - Significant with many small files
- New TCP connections start slow
  - Jacobson‘s slow start algorithm (cf. Rechnernetze)

FTP – Sample Session

```
220 (vsFTPd 2.3.5)
USER anonymous
331 Please specify the password.
PASS ***********
230 Login successful.
PWD
257 ""
    [Get directory]
TYPE A
200 Switching to ASCII mode.
PASV
227 Entering Passive Mode (132,252,181,48,164,166).
LIST
150 Here comes the directory listing.
    [Download
    Waiting for server...]
226 Directory send OK.
```

FTP – Sample Session (2)

```
CWD debian
250 Directory successfully changed.
PWD
257 "/debian"
    [Get directory]
PASV
227 Entering Passive Mode (132,252,181,48,205,31).
LIST
150 Here comes the directory listing.
    [Download
    Waiting for server...]
226 Directory send OK.
TYPE I
200 Switching to Binary mode.
PASV
227 Entering Passive Mode (132,252,181,48,207,107).
RETR README
150 Opening BINARY mode data connection for README (1495 bytes).
    [Download
    Waiting for server...]
226 Transfer complete.
```

FTP: Conclusion

- FTP is designed for reliable file transfers
- Suitable for local and wide area networks
- Outdated separation of data/control
  - Limits performance and connectivity
- User login but without encryption
  - Encapsulate in TLS
- Modern alternatives
  - SSH File Transfer Protocol
  - HTTPS and HTTP/2

UNIVERSITÄT DUISBURG ESSEN

Internet-Technologie & Web Engineering
Trivial File Transfer Protocol (TFTP)

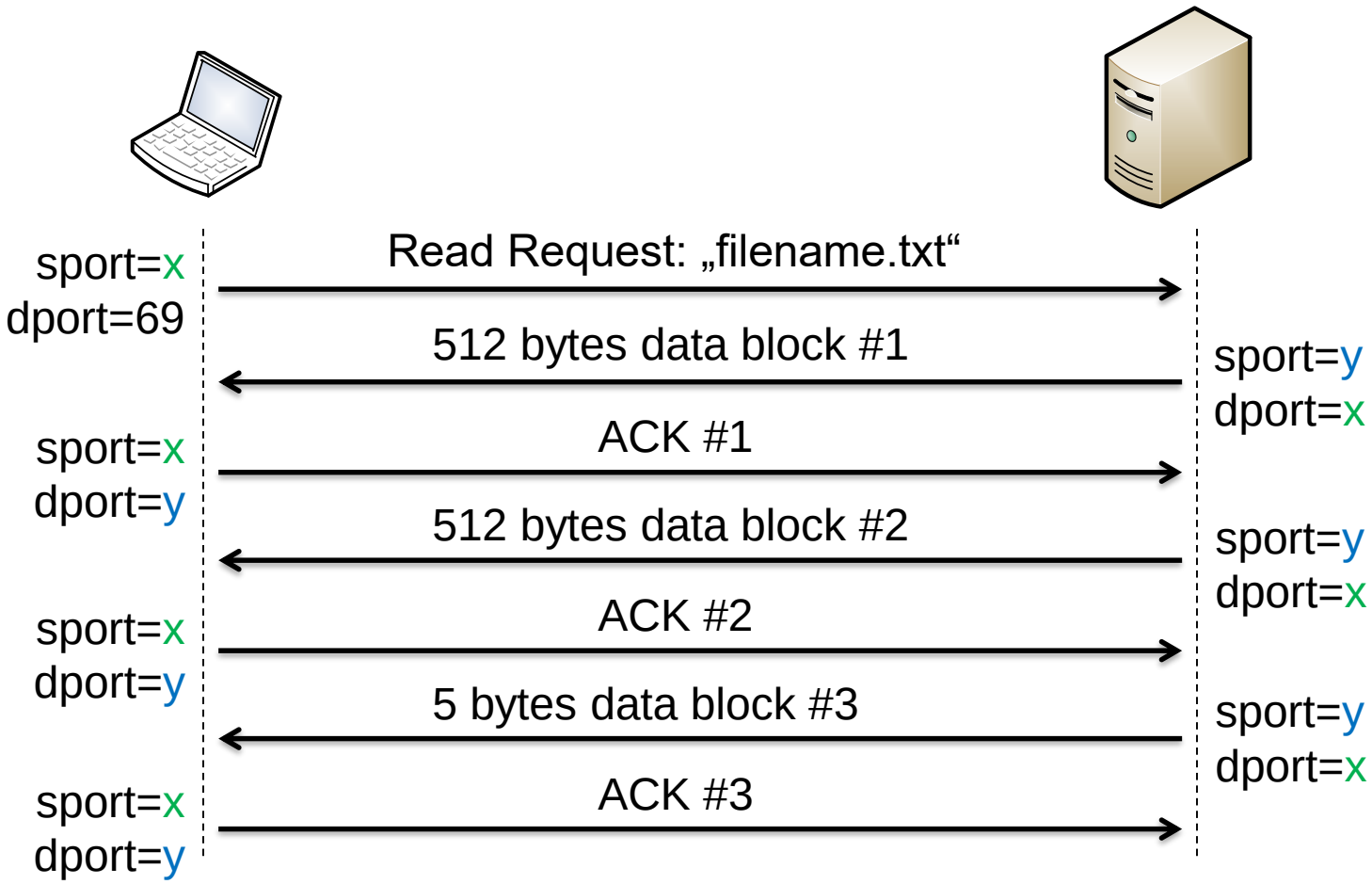
Dr.-Ing. Matthäus Wander

Universität Duisburg-Essen

TFTP: Overview

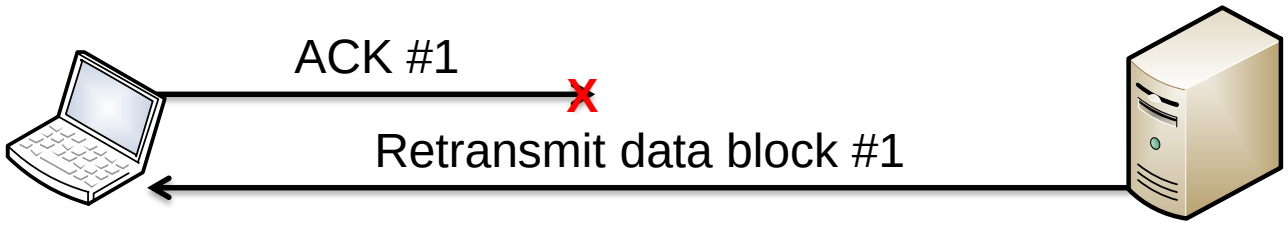
- IETF Internet Standard
  - Specified in RFC 1350
- Downgraded, simple version of FTP
  - No login, no authentication
  - No directory listings, no browsing
- Easy to implement
  - UDP transport only, no TCP
  - Good for tiny embedded systems or when booting operating system from network

File Download



UDP-based Network Protocol

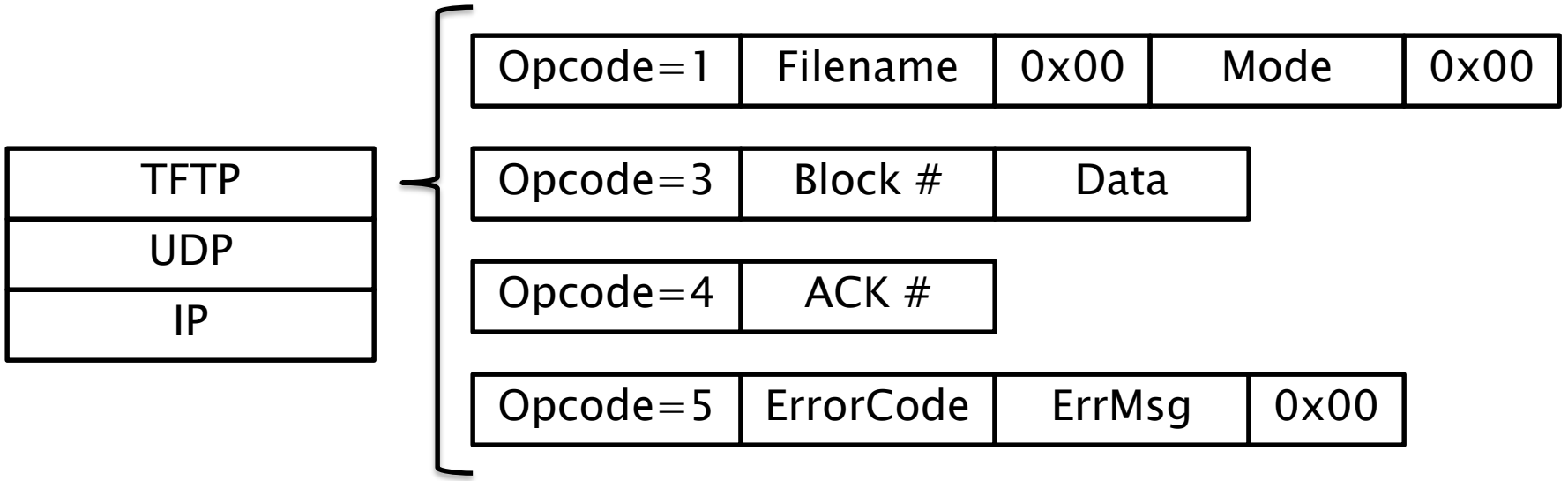
- Server listens on UDP destination port 69
  - Client uses source port x
  - Server responds with random source port y
  - x and y identify a session/connection/transaction
- UDP is unreliable: packet loss possible
  - Acknowledge every data packet
  - Retransmit in case of timeout



UDP-based Network Protocol (2)

- ACK provides flow control
  - Send one packet, wait for response
  - Simplex stop and wait (cf. Rechnernetze lecture)
- Max 512 bytes data per datagram (plus headers)
  - Slow transfer, especially with high latency
- TFTP is a stateful protocol
  - Client/server keep track of what was sent before and what they expect next (e.g. sent #4, next is #5)
  - But: state is very small

Message Format



- Opcode determines packet type
- Mode: text or 8-bit binary
  - In text mode, host converts machine-specific charset to standard ASCII

TFTP: Conclusion

- TFTP is simple and easy to implement
- UDP-based, but handles packet loss
- Stop and wait approach is slow
  - Especially with high latency
- Suitable for local network, not for wide area
- No security mechanisms