

**„Vorlesung Telematik“**  
(in English ☺)  
**Lecture: Computer Networks and the Internet**  
(Winter Semester 2004/05)

Prof. Dr. Dieter Hogrefe  
Dr. Xiaoming Fu  
Dipl.-Inf. Hannes Tschofenig (Siemens)

Telematics group  
University of Göttingen, Germany

---

Telematics group  
University of Göttingen, Germany

**Course Overview**

- Introduction
- Data link layer
- Network layer: routing, forwarding etc. and Mobile IP
- Transport layer and Quality of Service
- Application layer and multimedia networking
- Network security
- Advanced topics and course review
- Acknowledgements:
  - James Kurose, Keith Ross, Computer Networking(2nd Ed.), Addison-Wesley
  - Nick McKeown, Stanford University
  - Henning Schulzrinne, Columbia University
  - Y. Richard Yang, Yale University

WS 2004/05

---

Telematics group  
University of Göttingen, Germany

**Literatures**

- A. S. Tanenbaum, "Computer Networks", 4th edition, Prentice Hall, 2002. (main textbook)
- J. Kurose and K. Ross, "Computer Networking: A Top-Down Approach Featuring the Internet", 2nd edition, Addison-Wesley, 2002. (alternative main textbook)
- Jochen Schiller, "Mobile Communications", 2nd edition, Addison-Wesley, 2003
- J. Ellsberger, D. Hogrefe, A. Sarma, "SDL - Formal Object-Oriented Language for Communicating Systems", Prentice Hall, 1997
- ... Other materials may be available during the lecture

WS 2004/05

---

Telematics group  
University of Göttingen, Germany

**Table of Content**

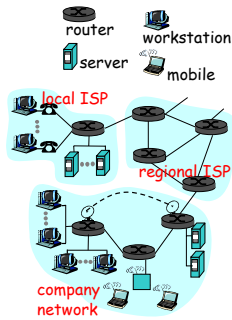
- Internet: an overview (we can watch a video together)
  - Basic components: *end-systems (hosts)*, *routers* and *communication links*
  - How to communicate: *protocols* defined by *standards*
  - Network edge: *connection-oriented* and *connectionless services*
  - Network core: *packet-switching* and *circuit switching*
  - An example of how TCP/IP works
- Network architectures
  - TCP/IP model
  - OSI model
- Specification of communication systems using SDL

WS 2004/05

---

### What's the Internet: "nuts and bolts" view

- millions of connected computing devices: *hosts, end-systems*
  - PCs workstations, servers
  - PDAs phones, toasters
  - running *network apps*
- *communication links*
  - fiber, copper, radio, satellite
  - transmission rate = **bandwidth**
- *routers*: forward packets (chunks of data)

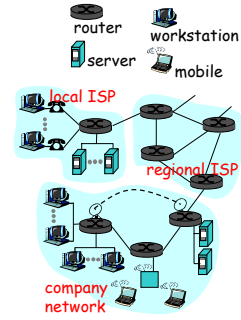


WS 2004/05

5

### What's the Internet: "nuts and bolts" view

- *protocols* define format, order of msgs sent and received among network entities, and actions taken on msg transmission, receipt
  - e.g., TCP, IP, HTTP, FTP, PPP
- *Internet: "network of networks"*
  - loosely hierarchical
  - public Internet versus private intranet
- Internet standards
  - RFC: Request for Comments
  - IETF: Internet Engineering Task Force

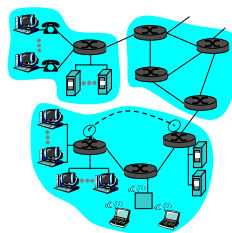


WS 2004/05

6

### What's the Internet: a service view

- *communication infrastructure* enables distributed applications:
  - Web, email, games, e-commerce, database., voting, file (MP3) sharing
- *communication services provided to apps*:
  - connectionless
  - connection-oriented
- *cyberspace* [Gibson]:
  - "a consensual hallucination experienced daily by billions of operators, in every nation, ...."

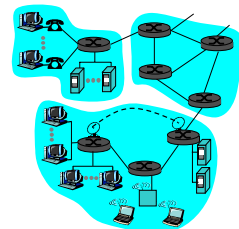


WS 2004/05

7

### A closer look at network structure:

- *network edge*: applications and hosts
- *network core*:
  - routers
  - network of networks
- *access networks, physical media*: communication links

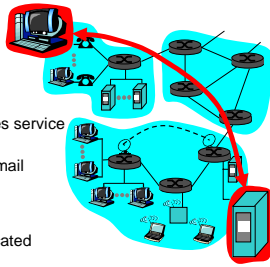


WS 2004/05

8

### The network edge:

- **end systems (hosts):**
  - run application programs
  - e.g. Web, email
  - at “edge of network”
- **client/server model**
  - client host requests, receives service from always-on server
  - e.g. Web browser/server; email client/server
- **peer-peer model:**
  - minimal (or no) use of dedicated servers
  - e.g. Gnutella, KaZaA



WS 2004/05

9

### Network edge: connection-oriented service

- Goal:** data transfer between end systems
- **handshaking:** setup (prepare for) data transfer ahead of time
    - Hello, hello back human protocol
    - **set up “state”** in two communicating hosts
  - TCP - Transmission Control Protocol
    - Internet’s connection-oriented service

### TCP service [RFC 793]

- **reliable, in-order** byte-stream data transfer
  - loss: acknowledgements and retransmissions
- **flow control:**
  - sender won’t overwhelm receiver
- **congestion control:**
  - senders “slow down sending rate” when network congested

WS 2004/05

10

### Network edge: connectionless service

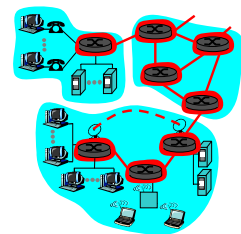
- Goal:** data transfer between end systems
- same as before!
  - **UDP - User Datagram Protocol [RFC 768]:** Internet’s connectionless service
    - unreliable data transfer
    - no flow control
    - no congestion control
- App’s using TCP:**
- HTTP (Web), FTP (file transfer), Telnet (remote login), SMTP (email)
- App’s using UDP:**
- streaming media, teleconferencing, DNS, Internet telephony

WS 2004/05

11

### The Network Core

- mesh of interconnected routers
- **the fundamental question:** how is data transferred through net?
  - **circuit switching:** dedicated circuit per call: telephone net
  - **packet-switching:** data sent thru net in discrete “chunks”



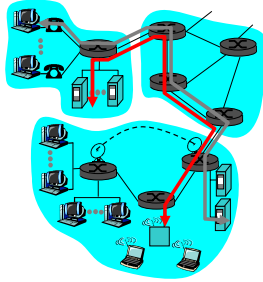
WS 2004/05

12

### Network Core: Circuit Switching

#### End-end resources reserved for "call"

- link bandwidth, switch capacity
- dedicated resources: no sharing, divided into "pieces":
  - frequency division (FDMA)
  - time division (TDMA)
- circuit-like (guaranteed) performance
- call setup required



### Network Core: Packet Switching

#### each end-end data stream divided into packets

- user A, B packets *share* network resources
  - Sequence of sending packets does not have fixed pattern → **statistical multiplexing**
- each packet uses full link bandwidth
- resources used *as needed*

#### resource contention:

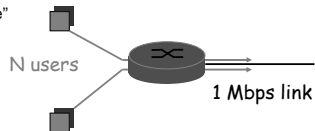
- aggregate resource demand can exceed amount available
- congestion: packets queue, wait for link use
- store and forward: packets move one hop at a time
  - transmit over link
  - wait turn at next link

~~Bandwidth division into "pieces"  
Dedicated allocation  
Resource reservation~~

### Packet switching versus circuit switching

Packet switching allows more users to use network!

- 1 Mbit link
- each user:
  - 100 kbps when "active"
  - active 10% of time
- circuit-switching:
  - 10 users
- packet switching:
  - with 35 users, probability > 10 active less than .0004



### Packet switching versus circuit switching

Is packet switching better than circuit switching?

- **Great for bursty data**
  - resource sharing
  - simpler, no call setup
- **However, in case of excessive congestion:** packet delay and loss
  - protocols needed for reliable data transfer, congestion control
- **Q: How to provide circuit-like behavior?**
  - bandwidth guarantees needed for audio/video apps
  - still an unsolved problem (will be discussed in part "Quality of Service")

### Packet-switching: store-and-forward



- Takes  $L/R$  seconds to transmit (push out) packet of  $L$  bits on to link of  $R$  bps
  - Entire packet must arrive at router before it can be transmitted on next link: **store and forward**
  - In this example, delay =  $3L/R$
- Example:**
- $L = 7.5$  Mbits
  - $R = 1.5$  Mbps
  - delay = 15 sec
- Note:**
- In order to be more efficient, large packets are usually segmented into smaller packets
- Can you explain why?

WS 2004/05

17

### Packet-switched networks: forwarding

- **Goal:** move packets through routers from source to destination
  - we'll study several path selection (i.e. routing) algorithms
- **datagram network:**
  - *destination address* in packet determines next hop
  - routes may change during session
  - analogy: driving, asking directions
- **virtual circuit network:**
  - each packet carries tag (virtual circuit ID), tag determines next hop
  - fixed path determined at *call setup time*, remains fixed thru call
  - *routers maintain per-call state*

WS 2004/05

18

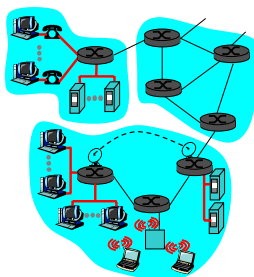
### Last hop to the end system: access network

**Q: How to connection end systems to edge router?**

- residential access nets
  - e.g., Cable modem, ADSL (asymmetric digital subscriber line)
- institutional access networks (school, company)
- mobile access networks

**Keep in mind:**

- bandwidth (bits per second) of access network?
- shared or dedicated?

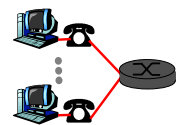


WS 2004/05

19

### Residential access: point to point access

- **Dialup via modem**
  - up to 56Kbps direct access to router (often less)
  - Can't surf and phone at same time: can't be "always on"
- **ADSL: asymmetric digital subscriber line**
  - up to 1 Mbps upstream (today typically < 256 kbps)
  - up to 8 Mbps downstream (today typically < 1 Mbps)
  - FDM: 50 kHz - 1 MHz for downstream  
4 kHz - 50 kHz for upstream  
0 kHz - 4 kHz for ordinary telephone



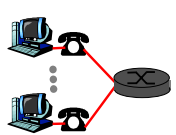
WS 2004/05

20

Telematics group  
University of Göttingen, Germany

### Residential access: point to point access

- **Dialup via modem**
  - up to 56Kbps direct access to router (often less)
  - Can't surf and phone at same time: can't be "always on"
- **ADSL: asymmetric digital subscriber line**
  - up to 1 Mbps upstream (today typically < 256 kbps)
  - up to 8 Mbps downstream (today typically < 1 Mbps)
  - FDM: 50 kHz - 1 MHz for downstream
    - 4 kHz - 50 kHz for upstream
    - 0 kHz - 4 kHz for ordinary telephone

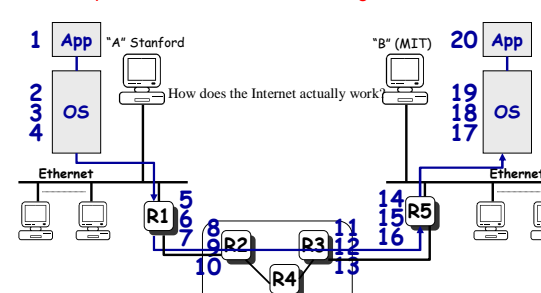


WS 2004/05 21

Telematics group  
University of Göttingen, Germany

### How does the Internet actually work?

- **Example: FTP over the Internet Using TCP/IP and Ethernet**

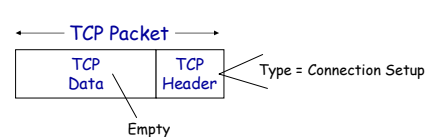


WS 2004/05 22

Telematics group  
University of Göttingen, Germany

### In the sending host (1)

1. **Application-Programming Interface (API)**
  - Application requests TCP connection with "B"
2. **Transmission Control Protocol (TCP)**
  - Creates TCP "Connection setup" packet
  - TCP requests IP packet to be sent to "B"

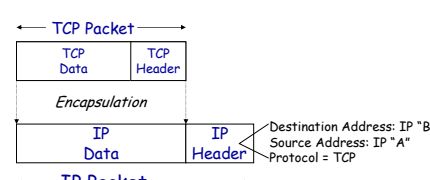


WS 2004/05 23

Telematics group  
University of Göttingen, Germany

### In the sending host (2)

3. **Internet Protocol (IP)**
  - Creates IP packet with correct addresses.
  - IP requests packet to be sent to router.



WS 2004/05 24

Telematics group  
University of Göttingen, Germany

### In the sending host (3)

**4. Link ("MAC" or Ethernet) Protocol**

- Creates MAC frame with Frame Check Sequence (FCS).
- Wait for Access to the line.
- MAC requests PHY to send each bit of the frame.

Destination Address: MAC "R1"  
Source Address: MAC "A"  
Protocol = IP

WS 2004/05 25

Telematics group  
University of Göttingen, Germany

### In Router R1 (1)

**5. Link ("MAC" or Ethernet) Protocol**

- Accept MAC frame, check address and Frame Check Sequence (FCS).
- Pass data to IP Protocol.

Destination Address: MAC "R1"  
Source Address: MAC "A"  
Protocol = IP

WS 2004/05 26

Telematics group  
University of Göttingen, Germany

### In Router R1 (2)

**6. Internet Protocol (IP)**

- Use IP destination address to decide where to send packet next ("next-hop routing").
- Request Link Protocol to transmit packet.

Destination Address: IP "B"  
Source Address: IP "A"  
Protocol = TCP

WS 2004/05 27

Telematics group  
University of Göttingen, Germany

### In Router R1 (3)

**7. Link ("MAC" or Ethernet) Protocol**

- Creates MAC frame with Frame Check Sequence (FCS).
- Wait for Access to the line.
- MAC requests PHY to send each bit of the frame.

Destination Address: MAC "R2"  
Source Address: MAC "R1"  
Protocol = IP

WS 2004/05 28

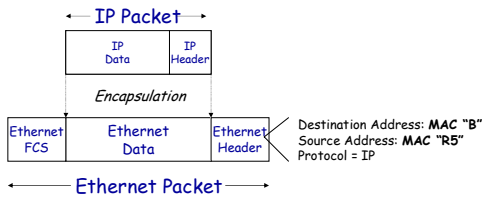
### In Routers R2, R3, R5

Same operations as Router R1

...after similar steps in 8-15:

#### 16. Link ("MAC" or Ethernet) Protocol

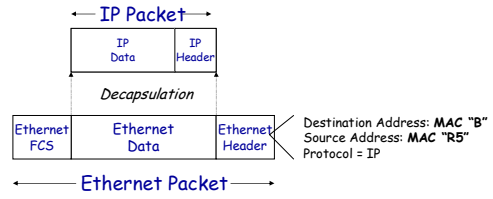
- Creates MAC frame with Frame Check Sequence (FCS).
- Wait for Access to the line.
- MAC requests PHY to send each bit of the frame.



### In the receiving host (1)

#### 17. Link ("MAC" or Ethernet) Protocol

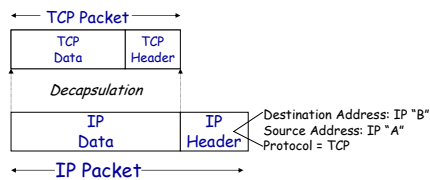
- Accept MAC frame, check address and Frame Check Sequence (FCS).
- Pass data to IP Protocol.



### In the receiving host (2)

#### 18. Internet Protocol (IP)

- Verify IP address.
- Extract/decapsulate TCP packet from IP packet.
- Pass TCP packet to TCP Protocol.



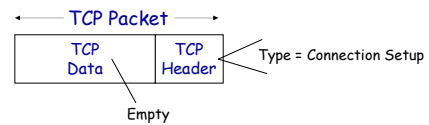
### In the receiving host (3)

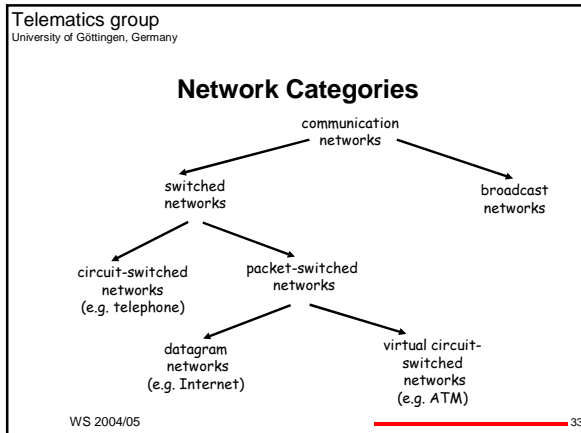
#### 19. Transmission Control Protocol (TCP)

- Accepts TCP "Connection setup" packet
- Establishes connection by sending "Ack".

#### 20. Application-Programming Interface (API)

- Application receives request for TCP connection with "A".





Telematics group  
University of Göttingen, Germany

### Network layering

Networks are complex!

- many "pieces":
  - hardware
    - hosts
    - routers
    - links of various media
  - software
    - applications
    - protocols

**Question:**  
Is there any hope of  
*organizing* the structure  
of networks?

Or at least our discussion  
of networks?

WS 2004/05 34

Telematics group  
University of Göttingen, Germany

### What is Layering?

- A technique to organize a network system into a **succession** of logically distinct entities, such that the service provided by one entity is **solely** based on the service provided by the previous (lower level) entity

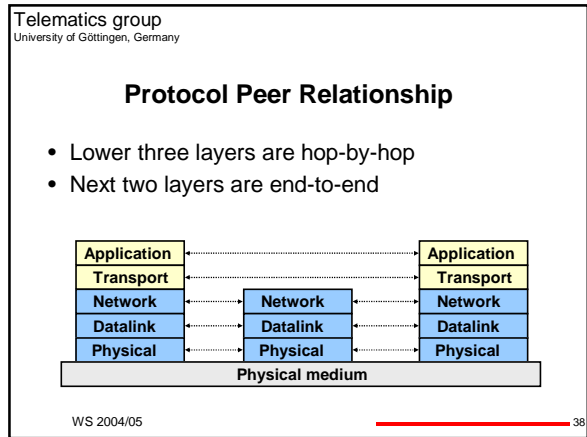
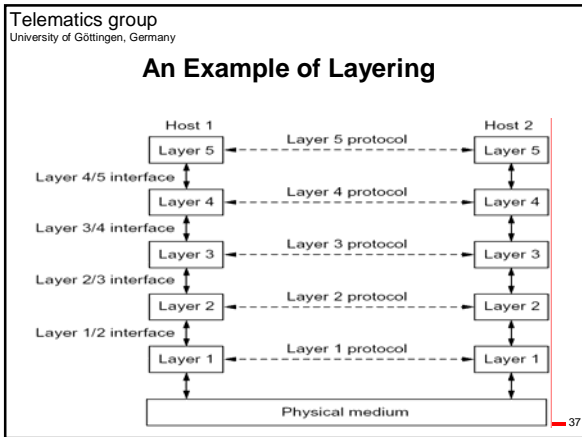
WS 2004/05 35

Telematics group  
University of Göttingen, Germany

### ISO/OSI Concepts

- ISO – International Standard Organization
- OSI – Open System Interconnection
- Service – says **what** a layer does
- Interface – says **how** to **access** the service
- Protocol – says **how** the service is **implemented**
  - a set of rules and formats that govern the communications between two **peers**

WS 2004/05 36



Telematics group  
University of Göttingen, Germany

### Why Layering?

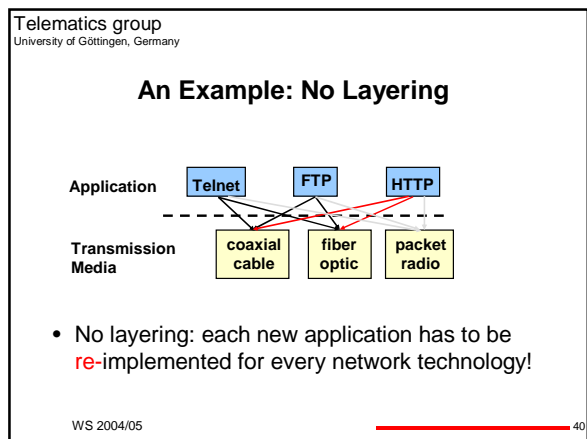
Dealing with complex systems:

- Explicit structure allows identification of the relationship among a complex system's pieces
  - layered **reference model** for discussion
- Modularization eases maintenance, updating of system
  - change of implementation of a layer's service transparent to rest of system
  - e.g., change in routing protocol doesn't affect rest of system

However, layering can lead to inefficient implementations

WS 2004/05

39



Telematics group  
University of Göttingen, Germany

### An Example: Benefit of Layering

- Solution: introduce an intermediate layer that provides a **common** abstraction for various network technologies

Application: Telnet, FTP, HTTP

Transport & Network

Transmission Media: coaxial cable, fiber optic, packet radio

WS 2004/05 41

Telematics group  
University of Göttingen, Germany

### ISO OSI Reference Model

- Seven layers
  - lower three layers are hop-by-hop
  - next four layers are end-to-end

Application, Presentation, Session, Transport, Network, Datalink, Physical

Physical medium

WS 2004/05 42

Telematics group  
University of Göttingen, Germany

### TCP/IP Architecture: Internet Layering

- Lower three layers are hop-by-hop
- Next two layers are end-to-end

Application, Transport, Network, Datalink, Physical

Physical medium

WS 2004/05 43

Telematics group  
University of Göttingen, Germany

### Example: Internet Protocol Layers

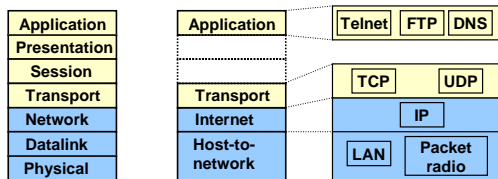
- **Five layers**
  - **Application:** supporting network applications
    - ftp, smtp, http
  - **Transport:** host-host data transfer
    - tcp, udp
  - **Network:** routing of datagram from source to destination
    - ip, routing protocols
  - **Link:** data transfer between neighboring network elements
    - ppp, ethernet
  - **Physical:** bits "on the wire"

application, transport, network, link, physical

WS 2004/05 44

### Internet Layering and OSI Layering

- OSI: conceptually define: service, interface, protocol
- Internet: provide a successful implementation



### The “End-to-End Argument”

- If the application can do it, don't do it at a lower layer -- the application knows the best what it needs
  - add functionality in lower layers *iff* it
    - (1) is used and improves performance of a large number of (current and potential future) applications, and
    - (2) does not hurt (too much) other applications
- Success story: Internet

### Internet History

#### 1961-1972: Early packet-switching principles

- 1961: Kleinrock - queueing theory shows effectiveness of packet-switching
- 1964: Baran - packet-switching in military nets
- 1967: ARPAnet conceived by Advanced Research Projects Agency
- 1969: first ARPAnet node operational
- 1972:
  - ARPAnet demonstrated publicly
  - NCP (Network Control Protocol) first host-host protocol
  - first e-mail program
  - ARPAnet has 15 nodes

### Internet History

#### 1972-1980: Internetworking, new and proprietary nets

- 1970: ALOHAnet satellite network in Hawaii
- 1973: Metcalfe's PhD thesis proposes Ethernet
- 1974: Cerf and Kahn - architecture for interconnecting networks
- late70's: proprietary architectures: DECnet, SNA, XNA
- late 70's: switching fixed length packets (ATM precursor)
- 1979: ARPAnet has 200 nodes

**Cerf and Kahn's internetworking principles:**

- minimalism, autonomy - no internal changes required to interconnect networks
- best effort service model
- stateless routers
- decentralized control

**define today's Internet architecture**

## Internet History

### 1980-1990: new protocols, a proliferation of networks

- 1983: deployment of TCP/IP
- 1982: SMTP e-mail protocol defined
- 1983: DNS defined for name-to-IP-address translation
- 1985: FTP protocol defined
- 1988: TCP congestion control
- new national networks: Csnnet, BITnet, NSFnet, Minitel
- 100,000 hosts connected to confederation of networks

WS 2004/05

49

## Internet History

### 1990, 2000's: commercialization, the Web, new apps

- Early 1990's: ARPAnet decommissioned
- 1991: NSF lifts restrictions on commercial use of NSFnet (decommissioned, 1995)
- early 1990s: Web
  - hypertext [Bush 1945, Nelson 1960's]
  - HTML, HTTP: Berners-Lee
  - 1994: Mosaic, later Netscape
  - late 1990's: commercialization of the Web
- Late 1990's – 2000's:
  - more killer apps: instant messaging, peer2peer file sharing (e.g., Napster)
  - network security to forefront
  - est. 50 million host, 100 million+ users
  - backbone links running at Gbps

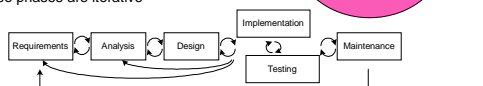
WS 2004/05

50

## Protocol Engineering: an Overview

Protocol engineering covers the whole life-cycle of a protocol

- requirements are set during overall system design: layering, behavior & communication
- protocol design produces complete specification
- specification needs to be verified
- protocol implementation is derived from specification
- implementation must be tested thoroughly
- these phases are iterative



WS 2004/05

51

## Protocol Design

How can we make a protocol design? Two approaches in practice:

- In ITU-T/ETSI, we generate precise specifications based on languages and tools, such as
  - SDL (Specification and Description Language)
  - MSC (Message Sequence Chart)
  - UML (Unified Modeling Language)
  - IDL (Interface Definition Language)
- Generate imprecise specs based on hobbyists and companies' consensus, no tools: IETF approach

Here we look at

**how to use SDL for protocol design**

WS 2004/05

52

Telematics group  
University of Göttingen, Germany

This part is partly derived from  
<http://www.netlab.hut.fi/opetus/s38157/s2003/>

### Specification of Communication Systems Using SDL

- SDL = Specification and Description Language
- Graphical and textual representation
- SDL specifications are based on Finite State Automations (FSA)
- FSA model of a protocol

(i) set of input elements  $I$  and set of output elements  $O$ , i.e. PDUs and Abstract Service Primitives (ASPs)

(ii) set of states  $S$  and state transition function  $succ$ :

$$succ \begin{cases} S \times I \rightarrow S & \text{(input transitions)} \\ S \times O \rightarrow S & \text{(output transitions)} \end{cases}$$

WS 2004/05 53

Telematics group  
University of Göttingen, Germany

### Overview of SDL

- SDL system is a model of a real world system
  - definition of system components
  - hierarchical model
  - Does not restrict the model, no absolute patterns
- a *System* consists of
  - *blocks*, that are connected with *channels*
- a *Block* consists of
  - *processes*, that are connected with signal routes
  - *sub-blocks*, that contain sub-blocks or processes
- a *Process* has
  - attributes defined with variables and external procedures
  - behaviours representing FSA, with states and transitions
- a *process* can contain certain *procedures*
- Types can be defined for
  - systems, blocks and processes

WS 2004/05 54

Telematics group  
University of Göttingen, Germany

### Structural Concepts

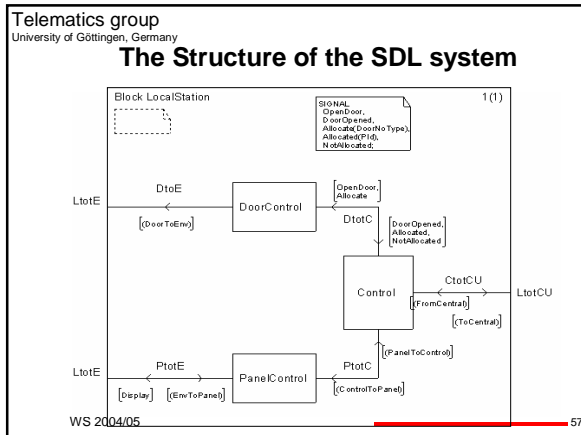
- SDL *system* constitutes the top level of detail
  - contains one or more blocks
    - Blocks interconnected with each other and with the boundary of the system by channels
    - Connections with outer system = openness of spec
  - system describes abstract machine communicating with env
  - a system diagram usually contains: system name, signal descriptions, channel descriptions, data type descriptions, block, descriptions and possibly (re-used packages)
- Channel is a means of conveying signals between blocks
  - Channels have direction
  - Can be bidirectional

WS 2004/05 55

Telematics group  
University of Göttingen, Germany

### The Structure of the SDL system

WS 2004/05 56

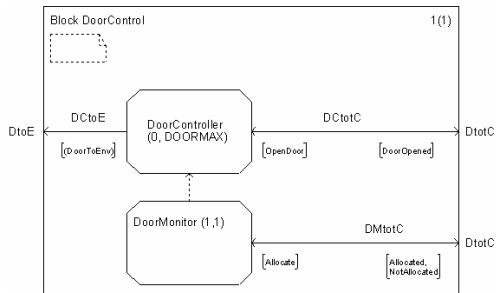


- Telematics group  
University of Göttingen, Germany
- ### A System and its Environment
- Channels usually are connected also to the environment
    - otherwise no external input/output to/from the system
  - Environment has processes
    - other parts of the real world system that are connected to the SDL system
      - e.g. users, large telecom system components
    - they are not specified with SDL
  - Environment observes and controls the system
    - user interfaces, key pads, etc.
- WS 2004/05

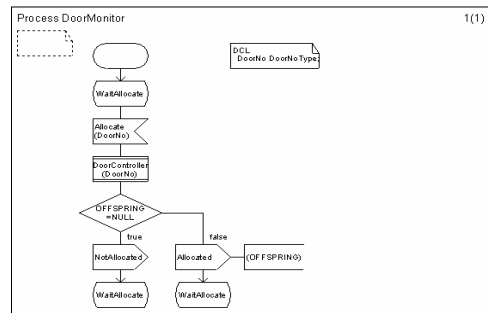
- Telematics group  
University of Göttingen, Germany
- ### Block
- System contains block(s)
  - The *block* is main structuring concept
    - can be partitioned into sub-blocks and channels
    - A block diagram usually contains the following elements:
      - block name
      - signal descriptions
      - Signal route descriptions
      - channel-to-route connections
      - process descriptions
  - Blocks do not themselves describe behaviour
  - Behaviour modelled through processes
- WS 2004/05

- Telematics group  
University of Göttingen, Germany
- ### Processes
- Basic components modelling the real world system
  - Processes have
    - attributes (data)
    - behaviour (state machine)
  - Processes communicate
    - with each other and with their environment
    - using signals (or other similar mechanisms)
  - Autonomous particles
  - Behaviour <-> Signalling
  - Signalling is asynchronous
  - Communication is defined by
    - signals
    - signal lists
    - signal routes
- 
- WS 2004/05

### The Structure of the SDL system



### Process Diagram



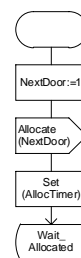
### Defining Behaviour

- Behaviour of the system can be divided in two
  - internal => processes
    - states and transitions
  - external => interacting with the environment
    - inputs and outputs
- Behaviour is defined using state machine language
  - Extended Finite State Machine (EFSM)
    - uses variables as an extension to 'state space'
  - Graphical notation

### Defining Behaviour

#### Initiating a process in start up (example)

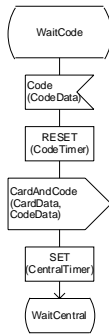
- start-up symbol
- task symbol
  - can be used e.g. for process initialisation
- output symbol
- timer start
- state symbol
  - first real state



## Defining Behaviour

### State transitions

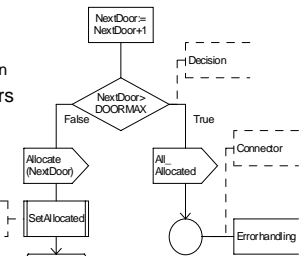
- input symbol
  - a signal from another process
  - a remote procedure call
- actions
  - tasks, output-symbols, timers, procedure calls, etc.
- new state
  - actual transition or to remain in the same state



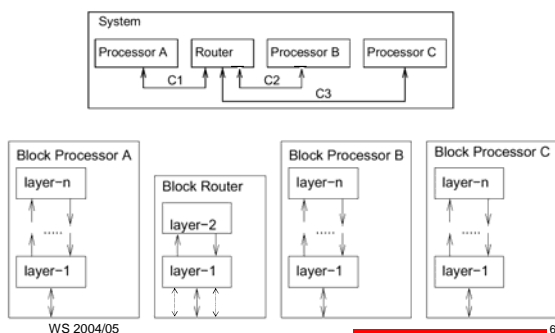
## Defining Behaviour

### Control structures

- decision symbol
  - evaluation => decision
- in- and out-connectors
- procedure calls and macro outlets
- enables 'normal' programming structures with



## Description of Network Protocols



## Introduction: Summary

Covered a "ton" of materials!

You now have:

- Internet overview
- Network edge & core
  - Connection-oriented v.s. connectionless services
  - Packet switching versus circuit switching
- An example for TCP/IP operation
- Layering, network architectures
  - OSI, TCP/IP
  - Internet history
- An introduction of protocol engineering and specification of communication systems in SDL