

Verteilte Systeme (Distributed Systems)

Karl M. Göschka

Karl.Goeschka@tuwien.ac.at

[http://www.infosys.tuwien.ac.at/teaching/courses/
VerteilteSysteme/](http://www.infosys.tuwien.ac.at/teaching/courses/VerteilteSysteme/)

Lecture 2:

Communication (Part 1)

- Networking Principles
- Internet: Ethernet and TCP/IP
- Middleware and adaptivity
- Remote procedure call

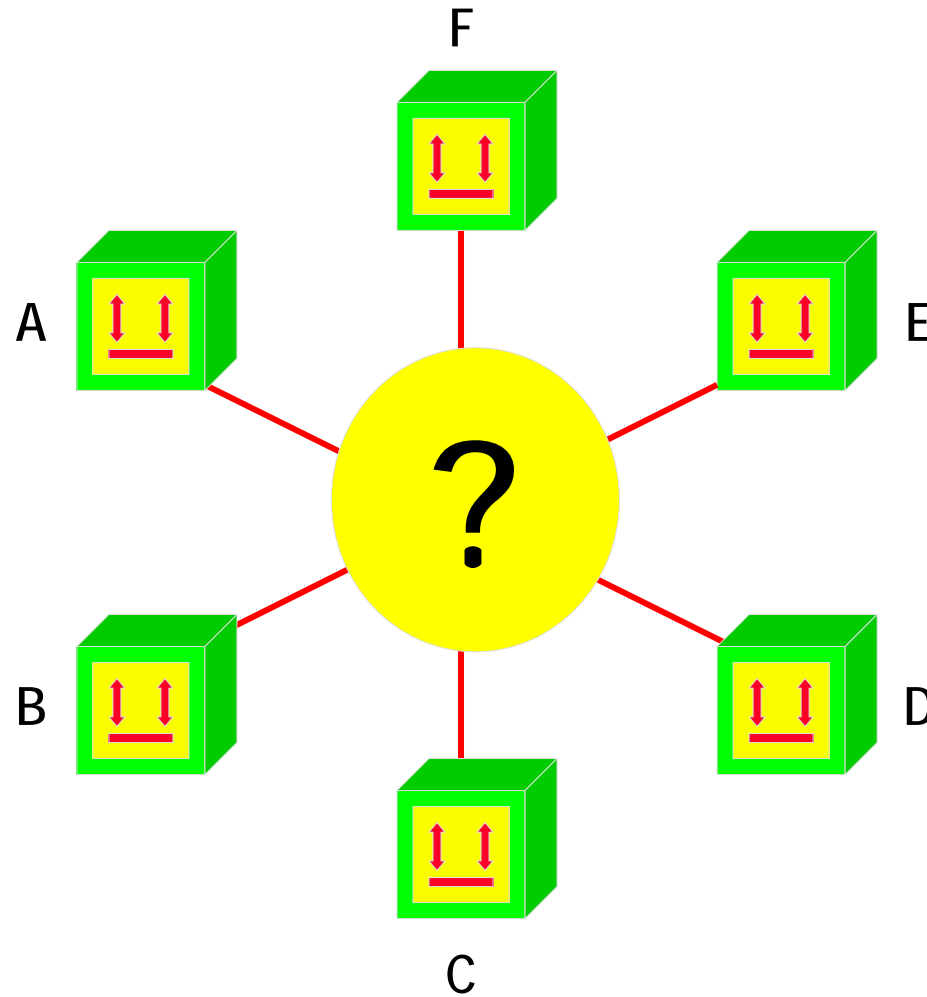
Communication requirements

- A distributed application consists of many entities (modules, objects, ...) that need to communicate
- Different applications require different properties from the communication (QoS)
- Example requirements:
 - Performance (latency and/or throughput)
 - Reliability
 - Continuous media
 - 1-1 or 1-n or n-n
 - Real-time
 - Consistent latency
 - Security

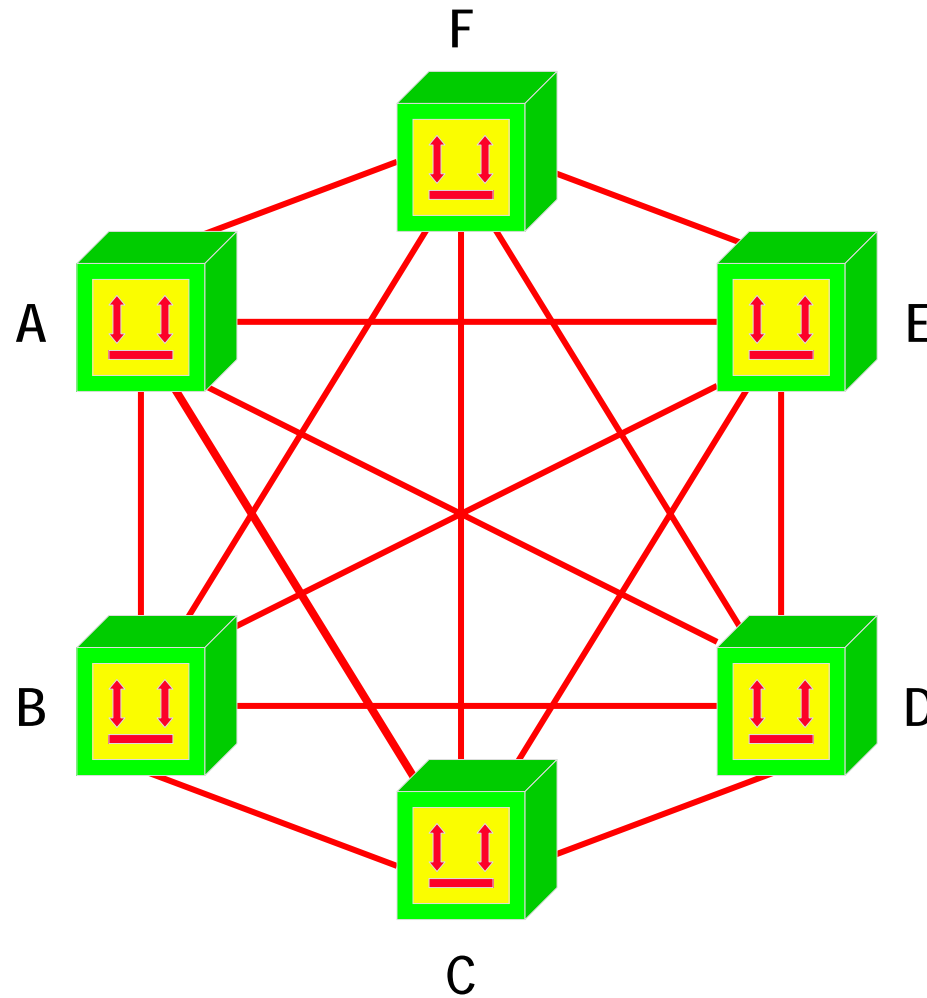
Communication model

- ❑ Communication system is responsible for providing communication (service)
- ❑ It consists of mechanisms (both hardware and software) for delivery of information from one entity to another
- ❑ Always based on low-level **message** passing!
- ❑ An entity is an **endpoint** of communication
- ❑ We need to identify (name) endpoints

How To Connect All Locations?



Any-to-Any Topology?



Networking Techniques

- any-to-any topology is very expensive
 - many lines are required and hence large number of transmission equipment (like modems, DSUs, line repeaters, etc.) necessary
 - many physical communication ports are required in devices
- to reduce costs
 - synchronous or asynchronous time division multiplexing principles can be used
 - circuit switching based on synchronous TDM
 - packet switching based on asynchronous (statistical) TDM

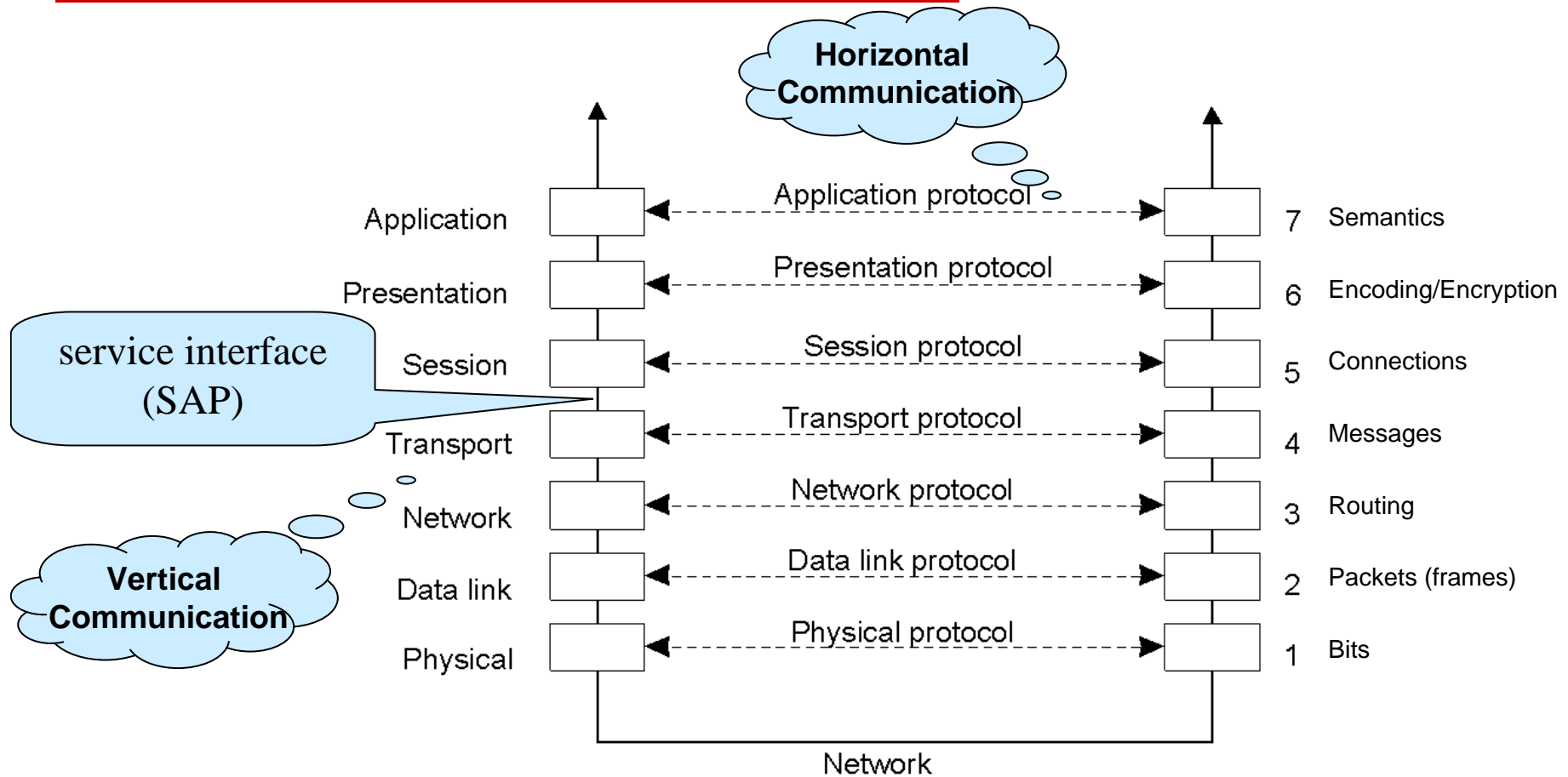
Idea of Layering and Services

- because communication between systems can be a very complex task
 - divide task of communication in multiple sub-tasks
 - so called layers
 - hence every **layer** implements only a part of the overall communication systems
- **hierarchically** organized
 - each layer receives services from the layer below
 - each layer serves for the layer above
- good for interoperability (needs **standards**)
 - capsulated entities and interfaces
- but increases (local) complexity

ISO/OSI Reference Model

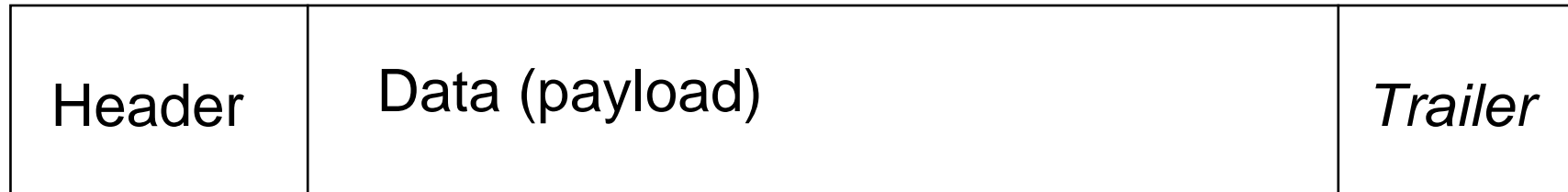
- where to define layers?
 - group functions (services) together
 - to allow changes in protocol, technology, and HW
 - to utilize existing protocols and HW
- layering of the ISO/OSI Reference Model
 - Open Systems Interconnection
 - defines tasks and interactions of seven layers
 - framework for development of communication standards
 - system-internal implementation is out of the scope
 - only external behavior of a system is defined by standards

ISO OSI 7 Layer Model



Layers, interfaces, and protocols in the OSI model.

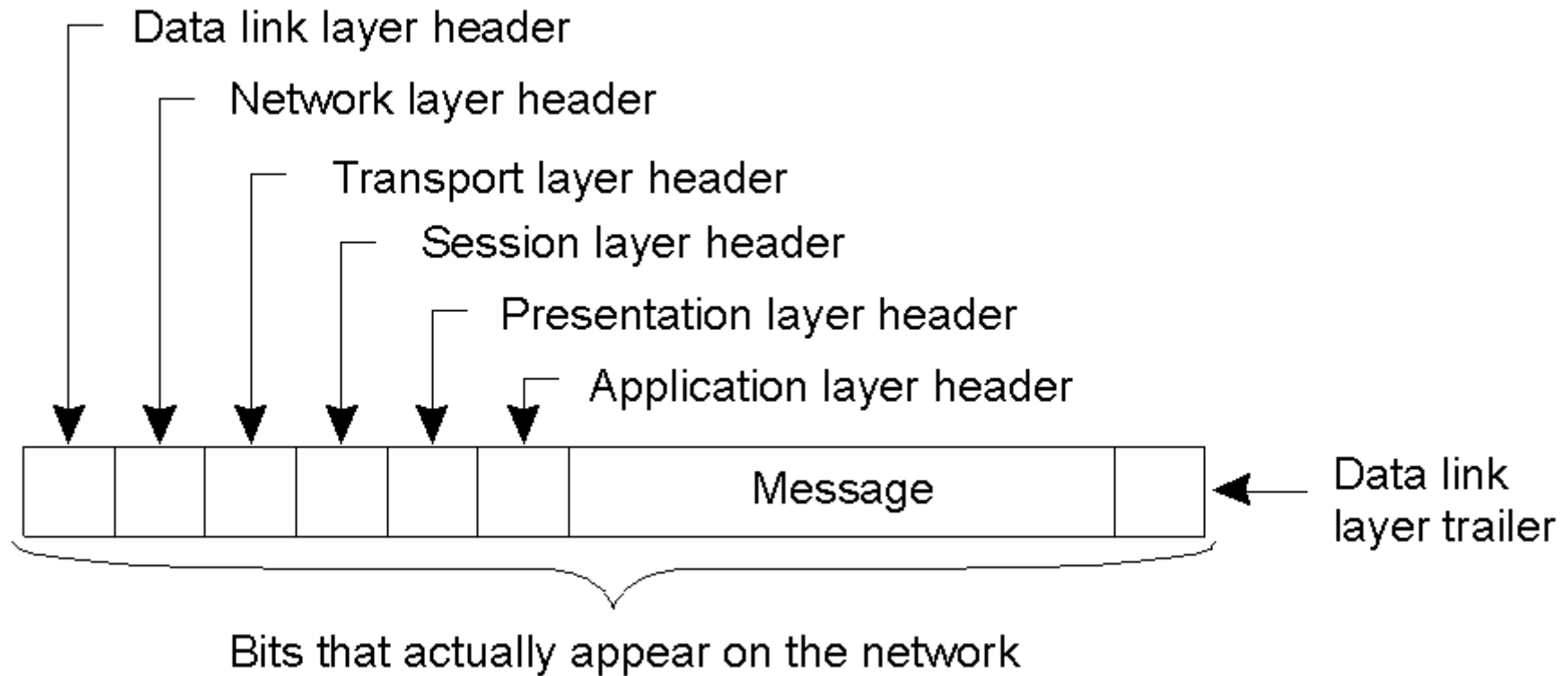
Encapsulation



Possible header contents:

- Message number*
- Message or header length*
- Priority*
- Checksum*
- Source and destination address*

Effect of Layered Protocols



A typical message as it appears on the network.

You Guess It...

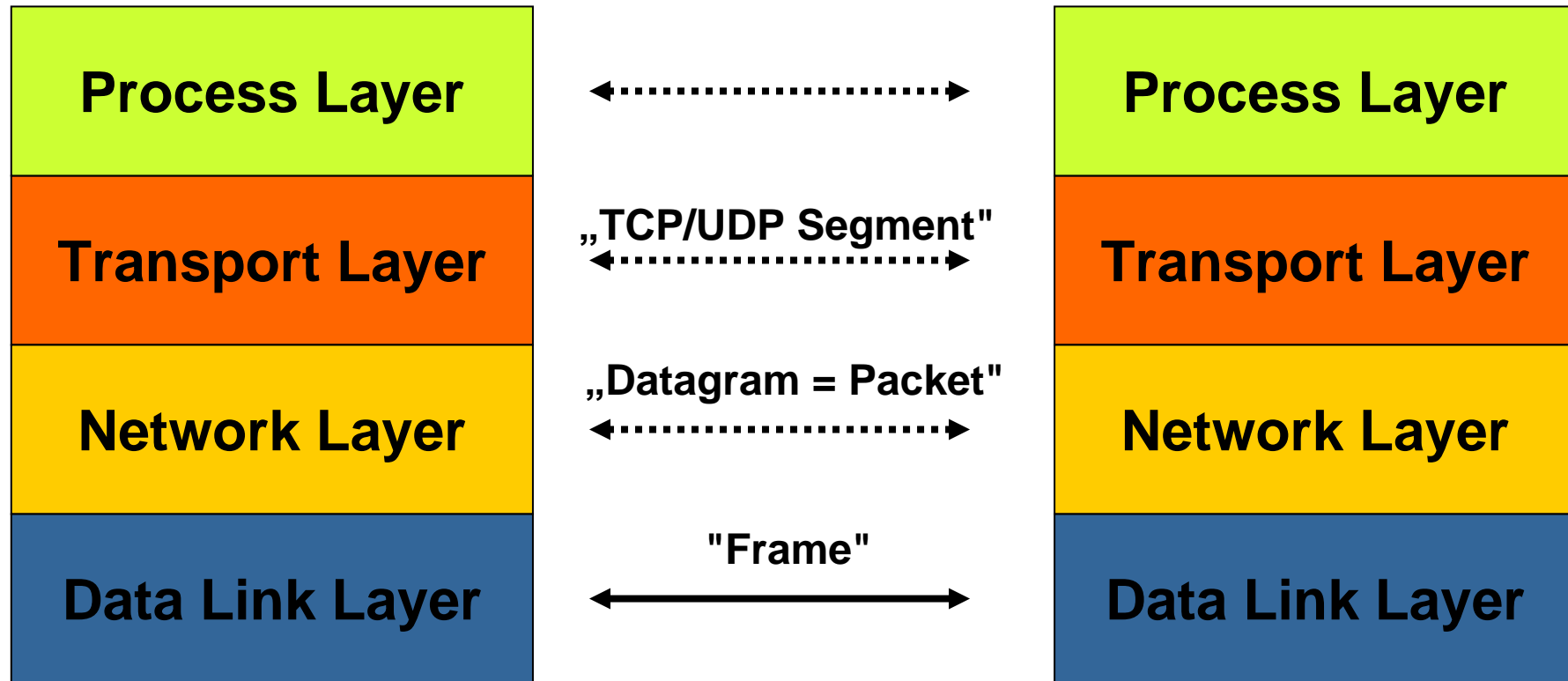
**Practical models
typically employ a
different number of
layers.**

Lecture 2:

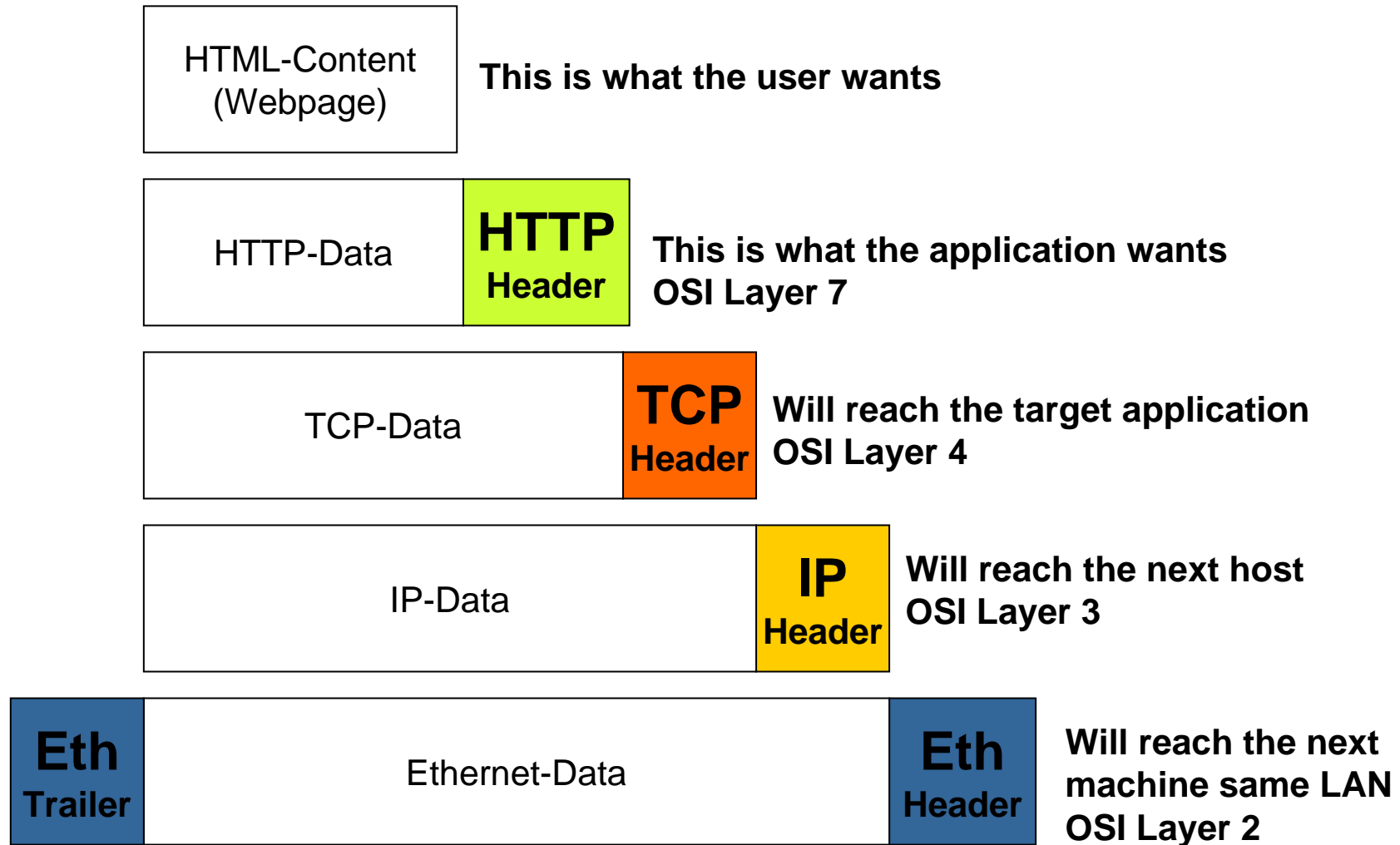
Communication (Part 1)

- Networking Principles
- Internet: Ethernet and TCP/IP
- Middleware and adaptivity
- Remote procedure call

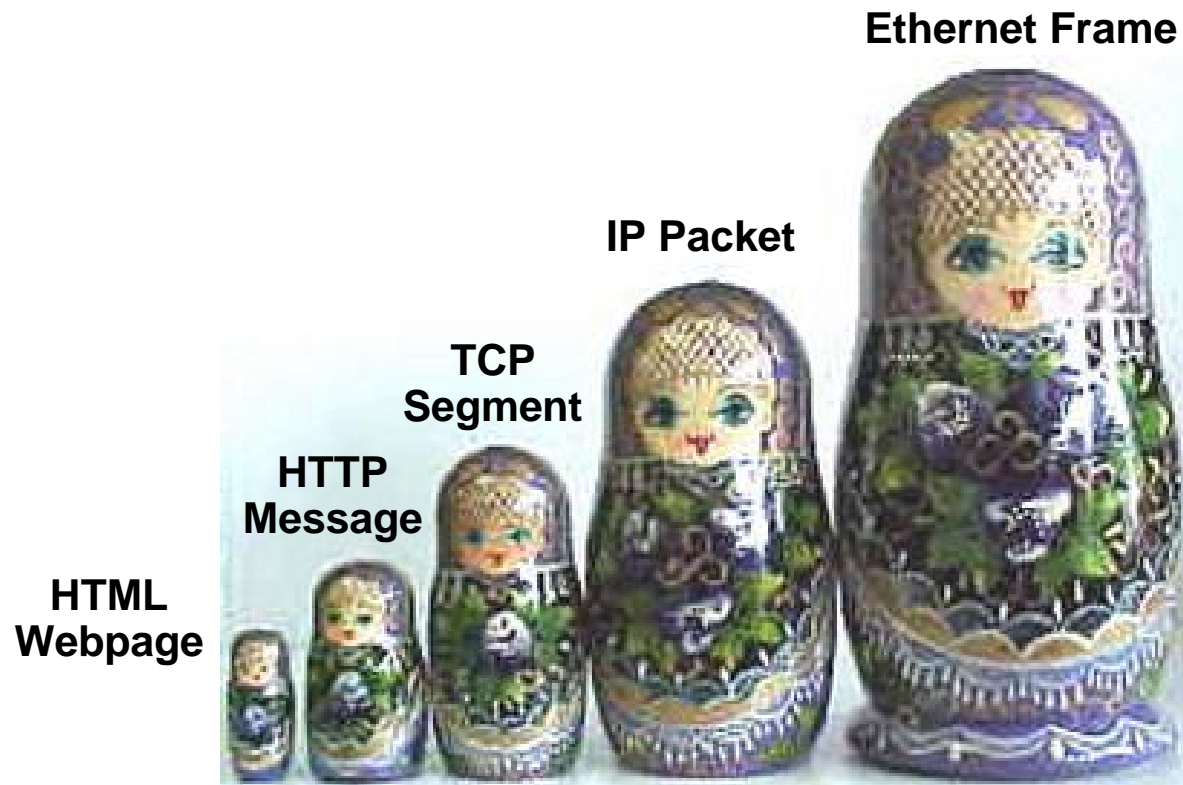
DoD 4-Layer Model (Internet)



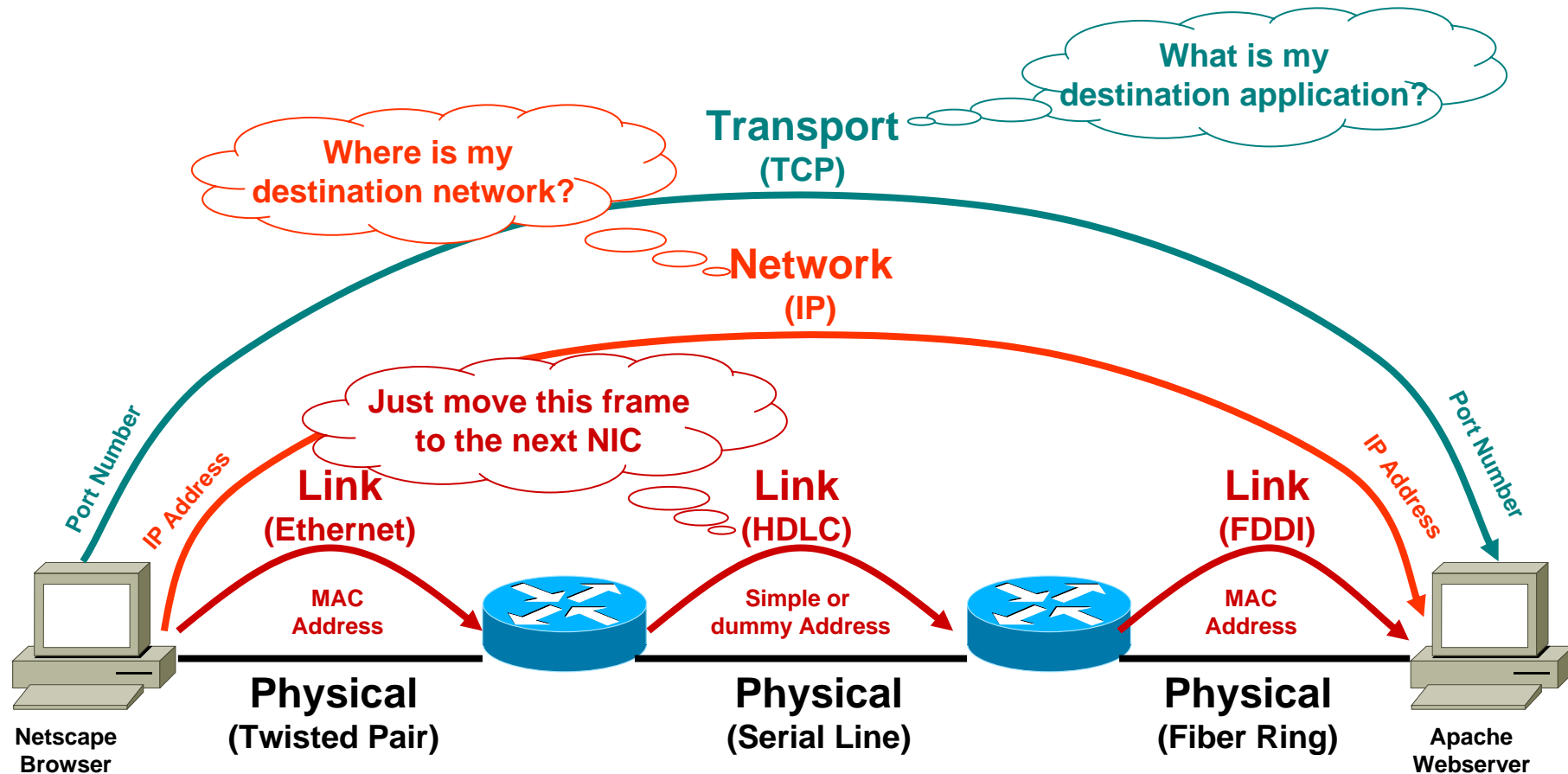
Internet Encapsulation



Practical Encapsulation



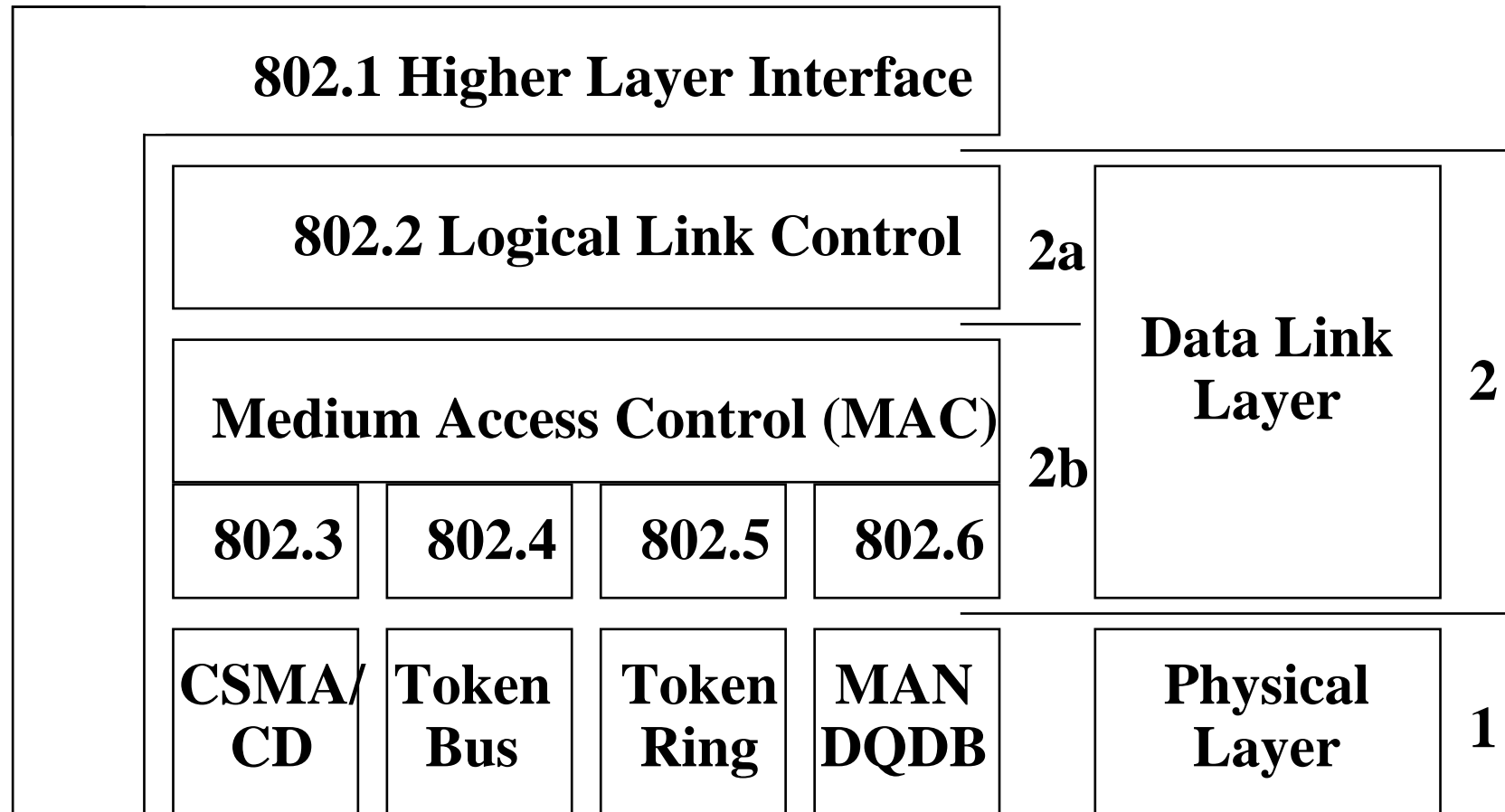
A Practical Example



IEEE 802 <-> OSI

IEEE 802

OSI



Principles of the Ethernet protocol

- ❑ Unique id for each node (**MAC address**)
- ❑ Distributed control — local decision-making (increases scalability)
- ❑ **Shared bus** (reduces scalability)
- ❑ CSMA/CD:
 - Carrier Sense
 - Multiple Access
 - Collision Detection

TCP/IP Protocol Suite

Application	FTP	SMTP	TELNET	TFTP	DNS	BOOTP		
Presentation								
Session							RIP	Rou- ting- Prot.
Transport	Transmission Control Protocol (TCP)			User Datagram Protocol (UDP)		OSPF		
Network	Internet Protocol (IP)							
	ICMP					ARP	RARP	
Link	IP Transmission über							
Physical	Ethernet RFC: 894	IEEE 802.x 1042	X.25 1356	SLIP 1055	PPP 1661			

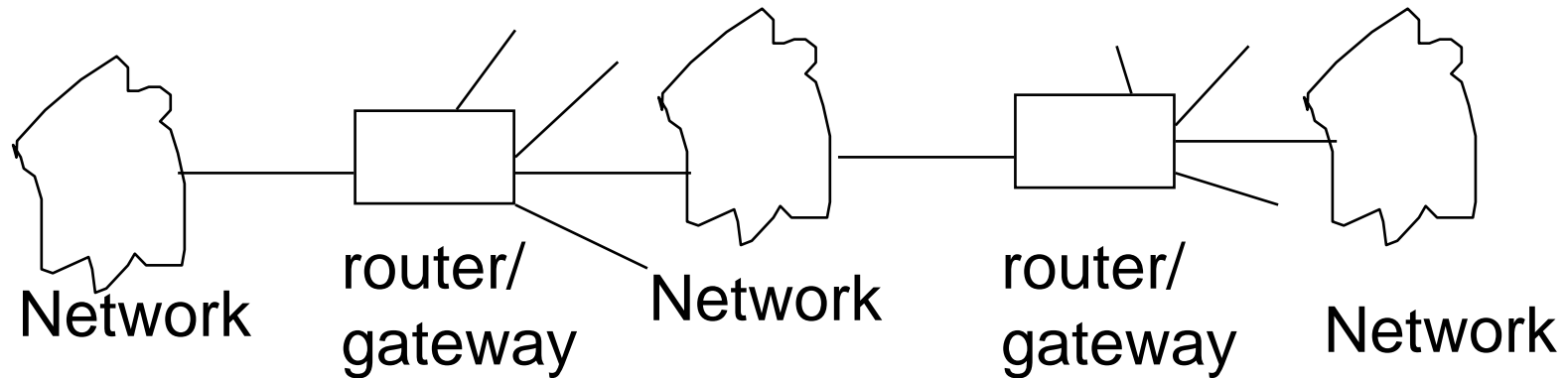
IP (Internet Protocol) principles

- ❑ RFC 791
- ❑ Virtual network
- ❑ Packet switched
- ❑ Connectionless (datagram)
- ❑ Transmission via several networks (**routing**)
- ❑ Unreliable (order, loss)

IP Protocol specifies

- Addressing
 - IP address: 32 bits (network id, host id)
 - IPv6: 128 bits (RFC 2460, 1998)
 - Must be unique for any node on the network (assigned by central authority)
- Packet format
 - Max packet size: 64k bytes
- Mechanisms for forwarding and routing
 - Based on network id, router sends packet to either local net or to a next router
- Fragmentation and Re-assembly
- Time to live (# hops)

IP routing using tables



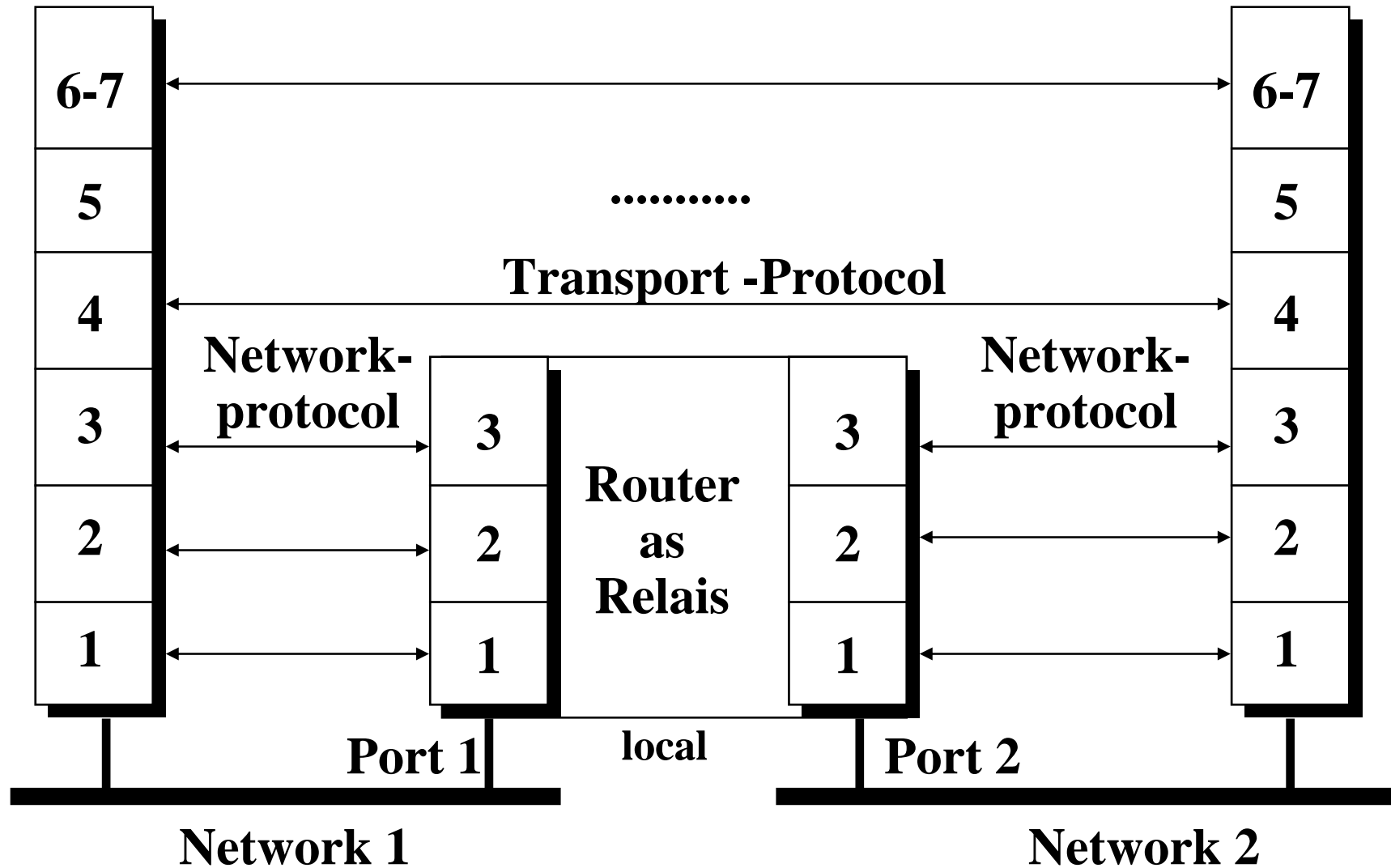
For network	Route to
128.0.0.0	deliver directly
250.44.0.0	128.12.0.5
.....

Remember that each packet has network id of **destination**

Router

System A

System B

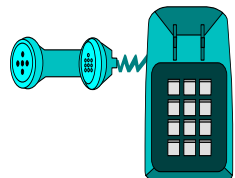


TCP/UDP transport

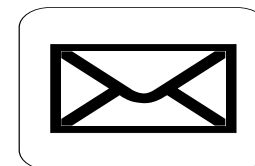
- ❑ TCP provides a reliable connection-oriented service; full-duplex; byte-streams
- ❑ UDP provides an unreliable, connection-less service
- ❑ Both use the services of IP
- ❑ Address is a pair (IP address, port number)
- ❑ Defines 1024 well-known ports
- ❑ No multicast or broadcast

Connection-oriented vs -less

- ❑ One partner calls
- ❑ Other partner accepts call
- ❑ Connection is established
- ❑ Communication starts
- ❑ One party terminates communication by closing the connection
- ❑ Reliability v. setup cost



- ❑ Sender writes a letter
- ❑ Puts address of receiver on letter
- ❑ Delivers letter to post office
- ❑ Receiver receives letter (eventually?)



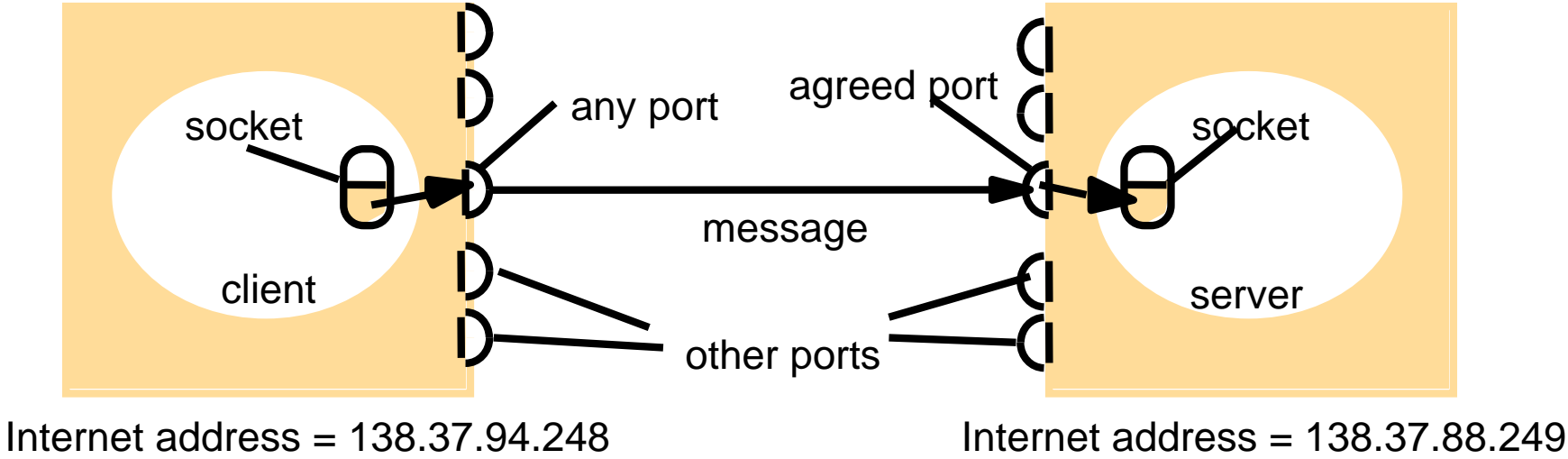
TCP principles

- ❑ RFC 793
- ❑ Connection-oriented service on layer 4
- ❑ **Addressing between processes** – no routing!
- ❑ **Reliable end-to-end** transport of data between processes: error detection and recovery regarding **order**, **loss**, **duplication**, and flow control (buffering)
- ❑ Provides network transparency for application
- ❑ addressing is through ports (assigned by OS)

UDP principles

- ❑ RFC 768
- ❑ Connectionless datagram service on layer 4
- ❑ Enhances layer 3 by forwarding to layer 7
- ❑ UDP addressing is through ports (assigned by OS)
- ❑ Not reliable with respect to loss of datagram (e.g. SNMP)
- ❑ Protocol implementation simple compared to TCP

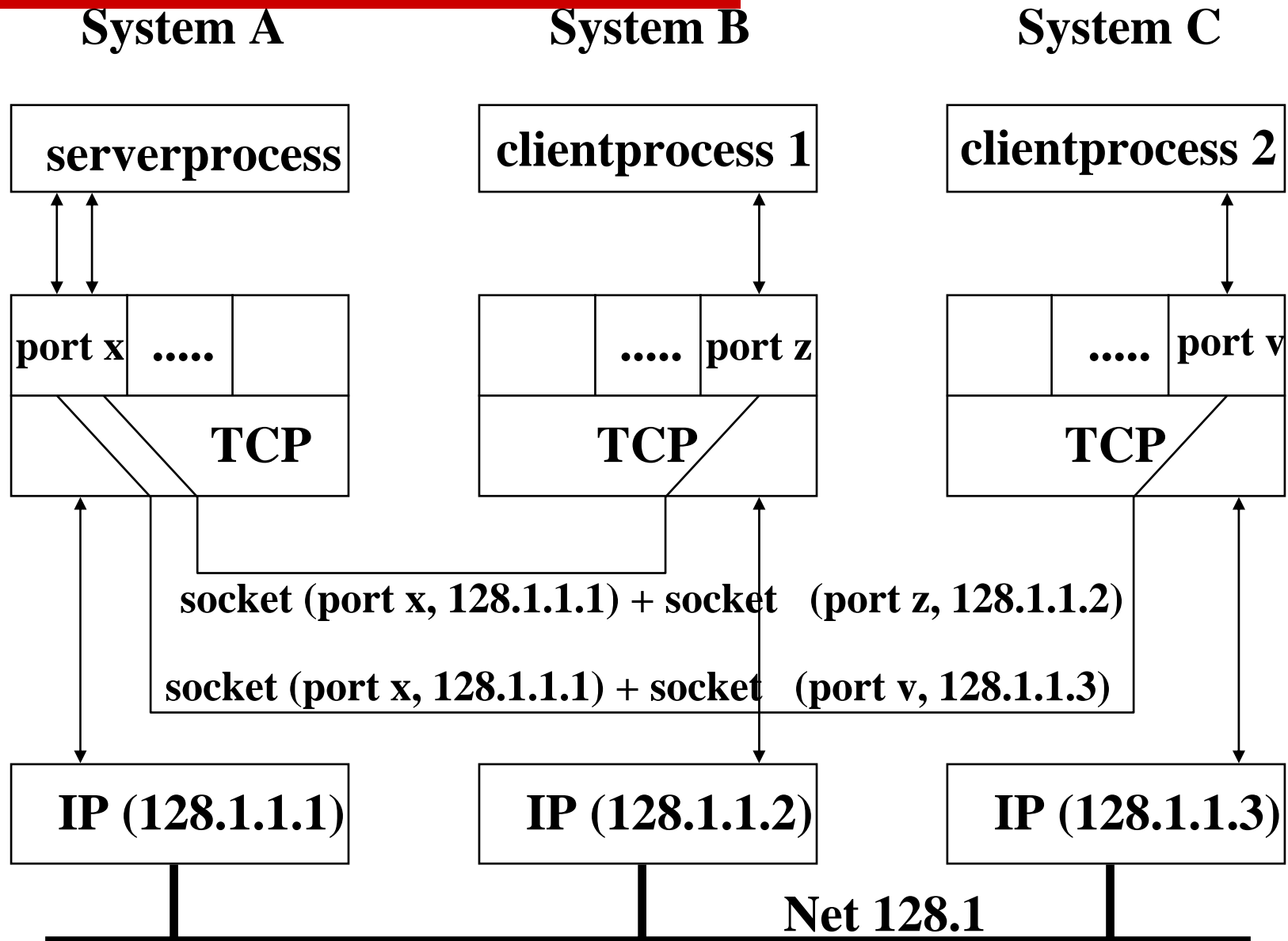
Sockets and ports



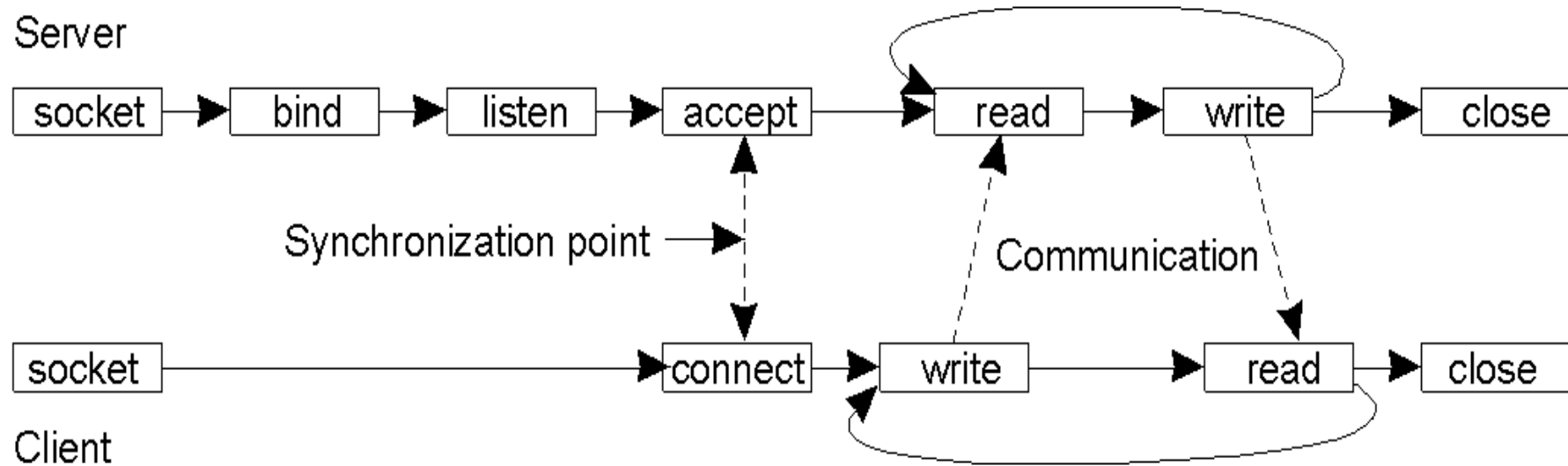
TCP Ports, Connections (3)

- ❑ Process needs to maintain several connections to other processes (e.g. client/server)
- ❑ → several virtual connections via one port (multiplexing), distinguished through **sockets**
- ❑ End point = IP address + port number
- ❑ TCP connection = endpoints of both partners
- ❑ Socket accesses connection

TCP Sockets, Connections (4)



Berkeley Sockets



Connection-oriented communication pattern using sockets.

Assignment of port numbers

- ❑ Ports on each node have integer numbers
- ❑ Applications may request ports dynamically
- ❑ Lower integers (0-1023) are reserved for “well-known” ports for server processes (e.g. time, rwho, telnet, http) (statically assigned)
- ❑ Thus, several and distinguishable connections to a server via a particular port

Three Way Handshake

- ❑ Unique initial values for sequence and acknowledgement numbers (synchronization for data transfer)
- ❑ Uses SYN and ACK flags plus respective sequence numbers
- ❑ “Old” invalid TCP segments can be detected and taken care of with RST flag

Data Transfer

- ❑ Positive acknowledgement: Each acknowledgement confirms the correctly (order!) received octets so far
- ❑ In case of a failure (packet order), the acknowledgement is suppressed until missing segment arrives, then explicit confirmation again
- ❑ Multiple acknowledgement: lost ACK is implicitly confirmed by next ACK (cumulative)
- ❑ Error correction: Timeout and re-transmission
- ❑ → Duplicate correction required

Summary: Communication protocols

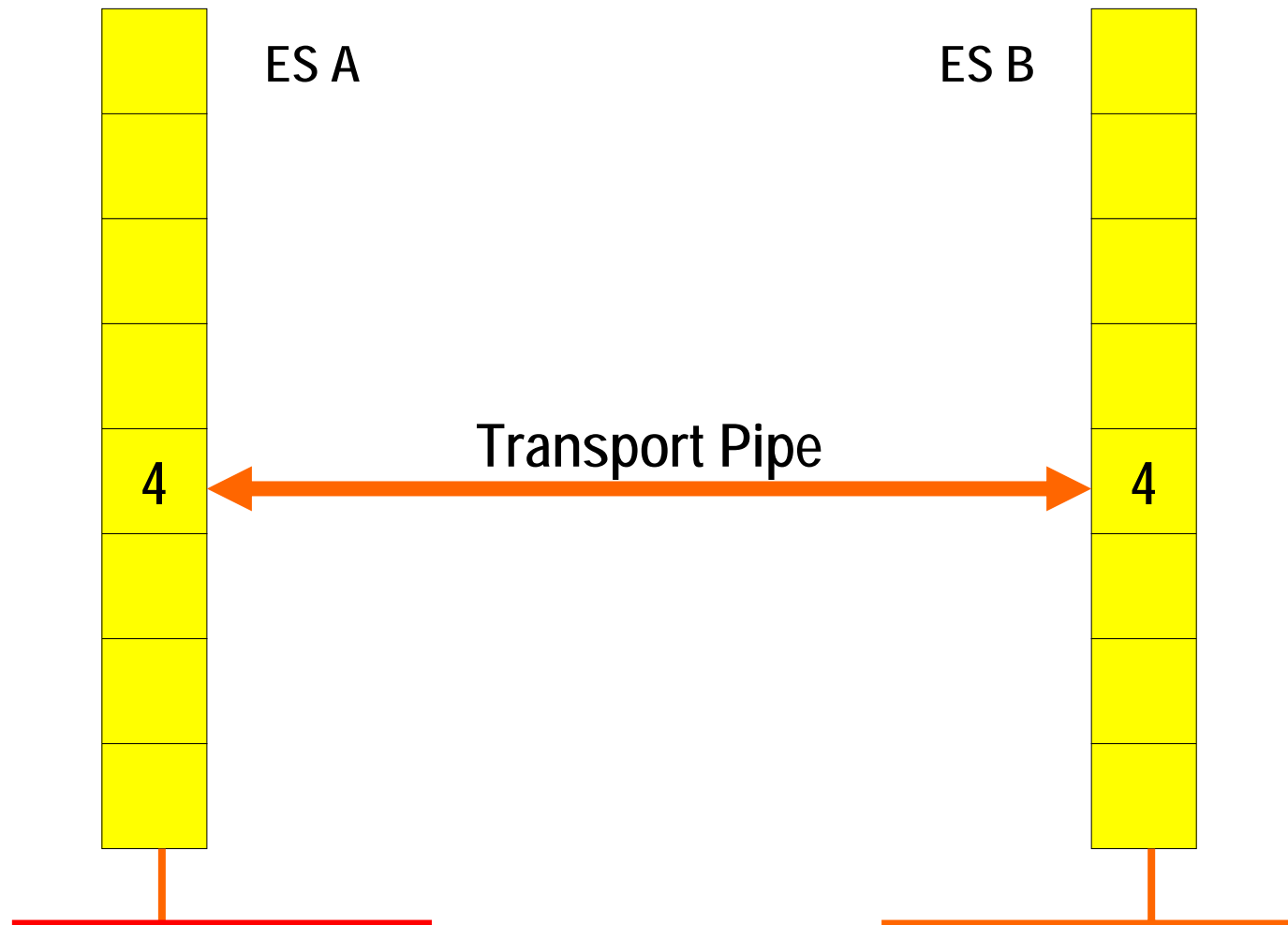
- ❑ Synchronous vs. asynchronous TDM
(circuit switched vs. packet switched)
- ❑ Connection-oriented vs. connection-less
(virtual path/call vs. datagram service)
- ❑ Reliable QoS vs. best-efforts
- ❑ Secure vs. insecure
- ❑ Synchronous vs. asynchronous message or
RPC
(blocking vs. non-blocking)
- ❑ Transient vs. persistent message passing
- ❑ Stateful vs. stateless

Lecture 2:

Communication (Part 1)

- Networking Principles
- Internet: Ethernet and TCP/IP
- Middleware and adaptivity
- Remote procedure call

How Layer 5 sees the Network



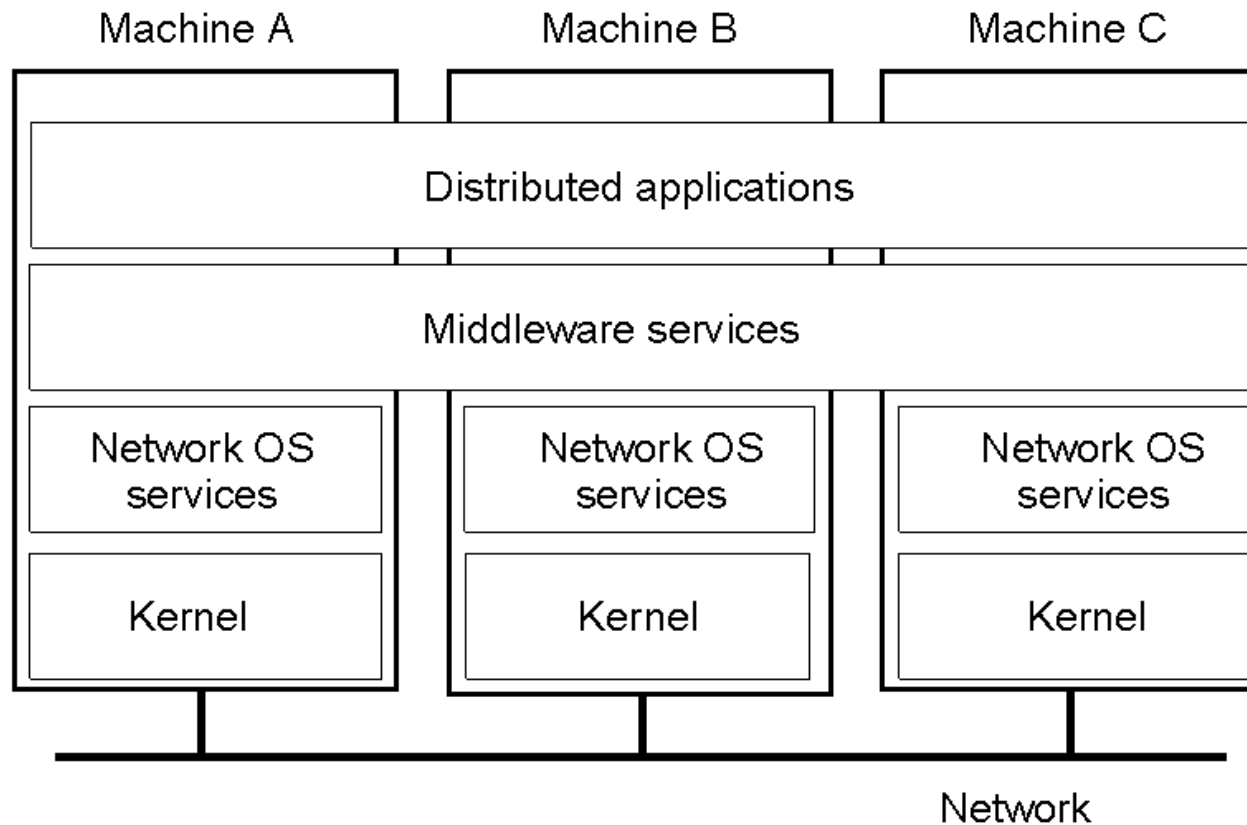
Above the transport layer?

- Typical enterprise applications consist of:
 - 70-80% application infrastructure
slight changes from app to app
 - 20-30% core business logic
different for every application

- → Middleware provides **run-time infrastructure** for applications

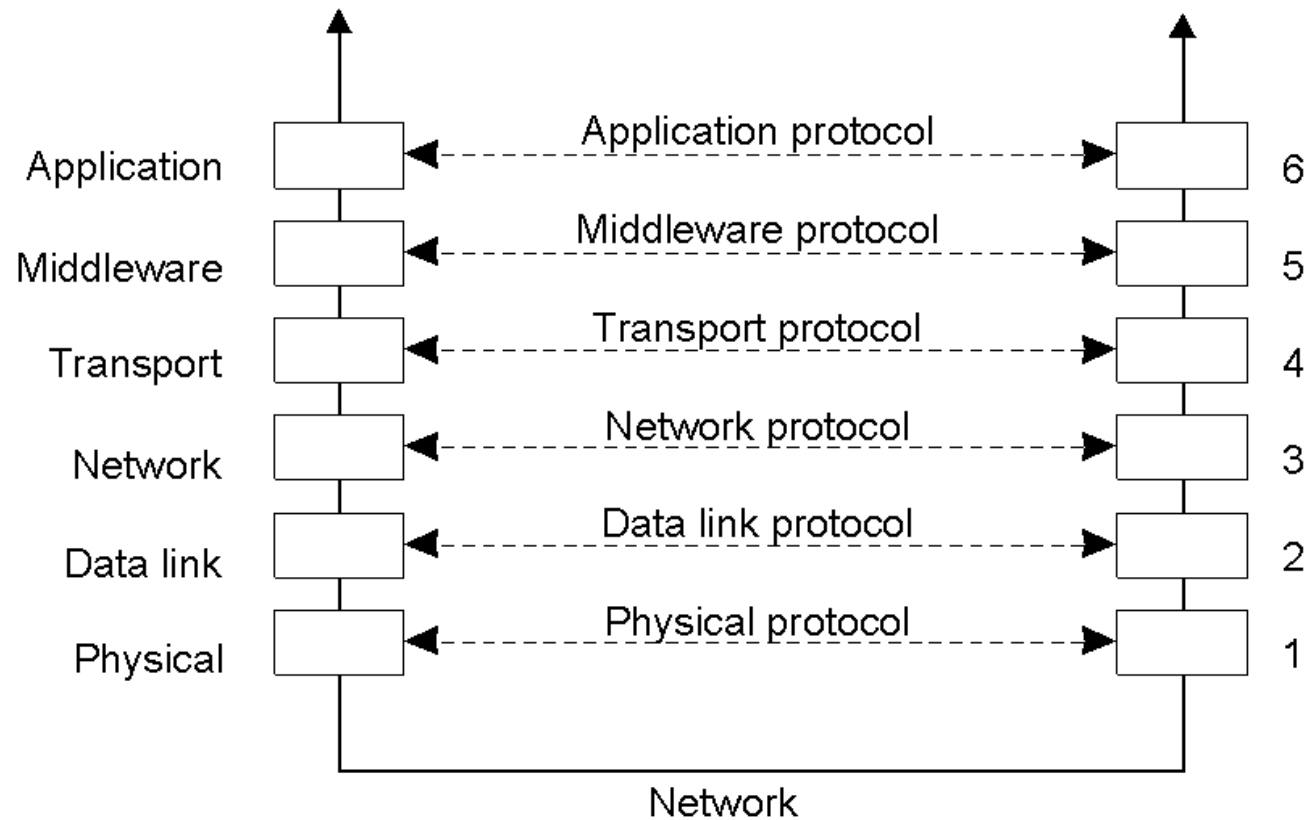
- → Middleware provides **programming abstraction** (and development tools)

Positioning Middleware



A distributed system organized as middleware.
Note that the middleware layer extends over multiple machines.

Middleware Protocols



An adapted reference model for networked communication.

Middleware requirements

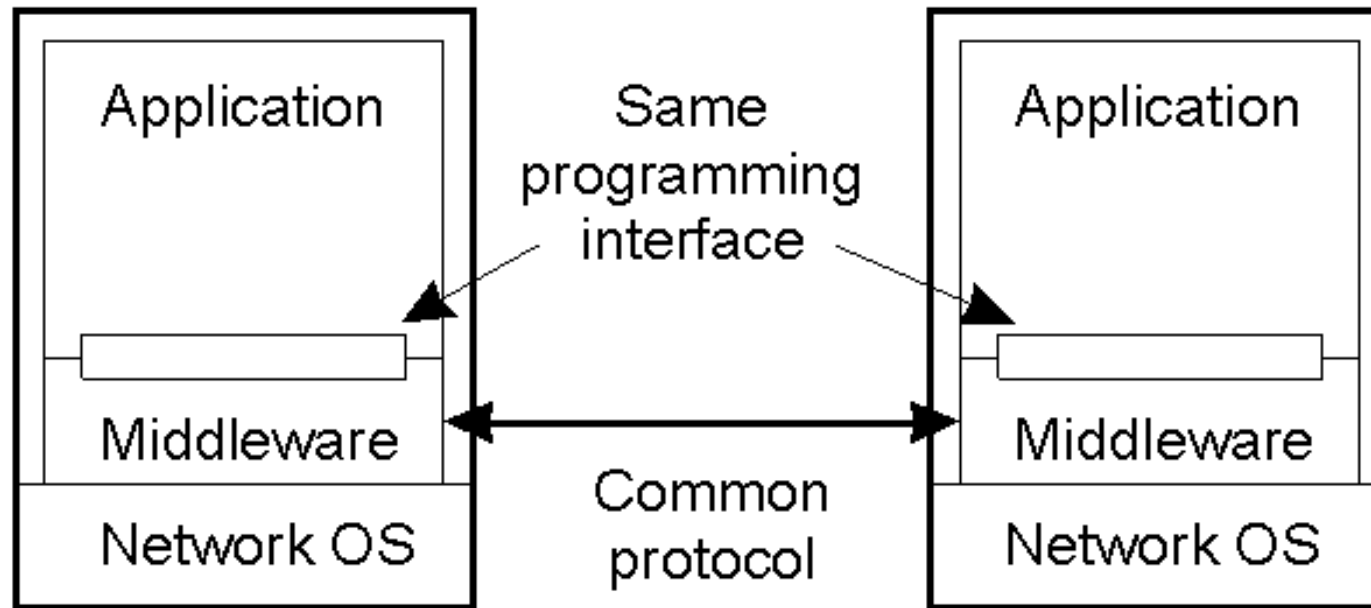
- ❑ Support transparency (with care)
- ❑ Support openness
- ❑ Provide means for scalability
- ❑ Support maintenance
- ❑ Life-cycle management of components
 - Activation/de-activation policies
- ❑ Support interface definition
- ❑ Provide communication primitives
- ❑ Provide support for concurrency control
 - Thread library

Middleware Services

- ❑ Communication facilities for access transparency
- ❑ Naming for location (and migration) transparency
- ❑ Persistence services or frameworks for persistence transparency (file, database)
- ❑ Distributed transactions for concurrency transparency; distributed locking
- ❑ Replication to assist failure transparency
- ❑ Security: Authentication, authorization

- ❑ → mainly concern extra-functional (quality) properties

Middleware and Openness



In an *open* middleware-based distributed system, the protocols used by each middleware layer, and the interfaces they offer to applications, should be the same.

Middleware and Architectural Styles

- Layered architectures (OSI)
- Object-based architectures (and components)
- Data-centered architectures (file based, database, resourceful WS, ...)
- Event-based architectures
- and combinations thereof

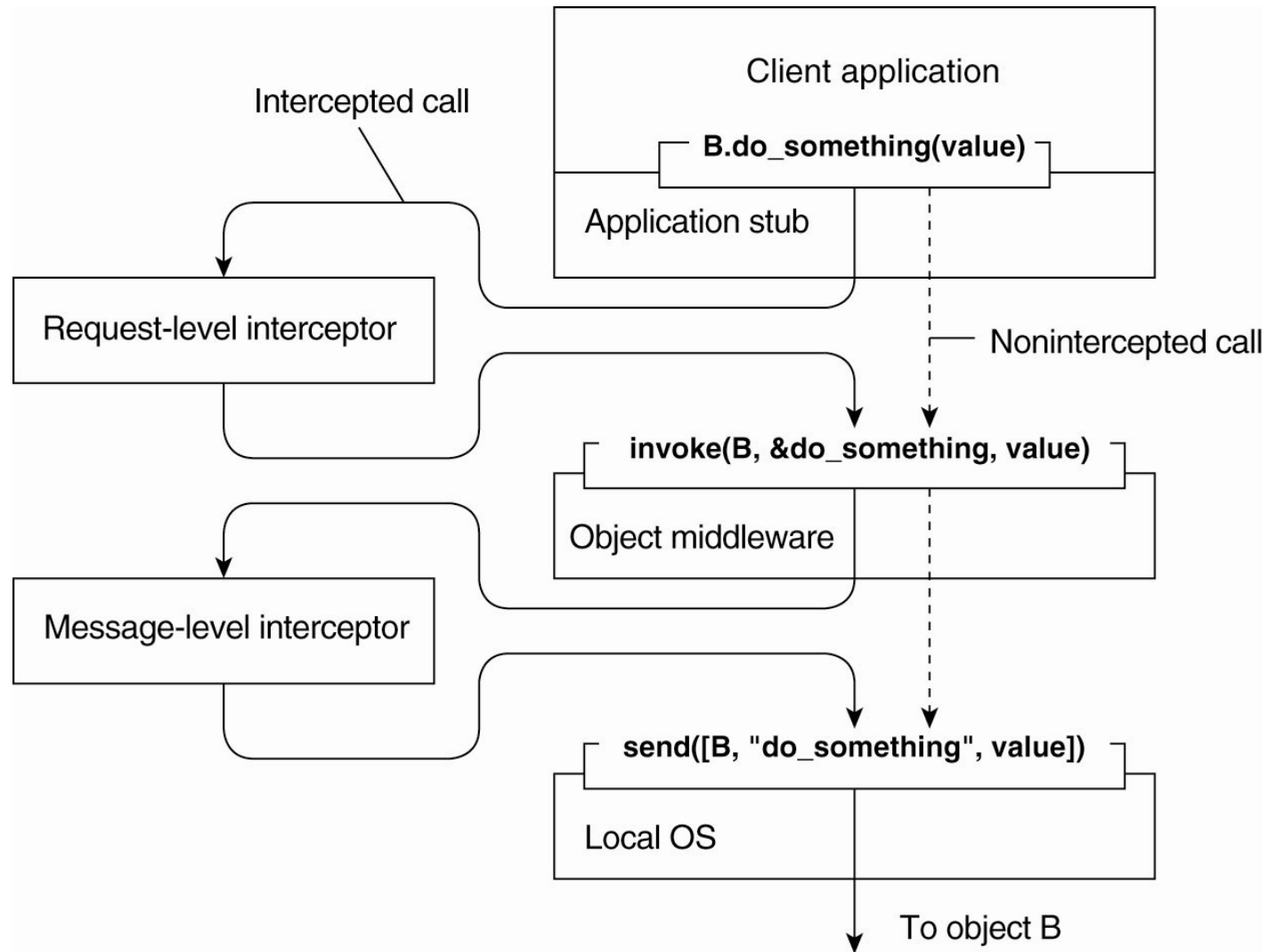
Middleware Models

- ❑ Everything is a file: no difference between local/remote, communication via shared files
- ❑ Distributed file system: transparency for traditional files only (but not for processes)
- ❑ Distributed Documents (WWW)
- ❑ Remote procedure calls RPC
- ❑ Remote method invocation RMI
- ❑ Message-oriented Middleware MoM
- ❑ Streams and continuous media (video server)

Enhancing the flexibility of MW

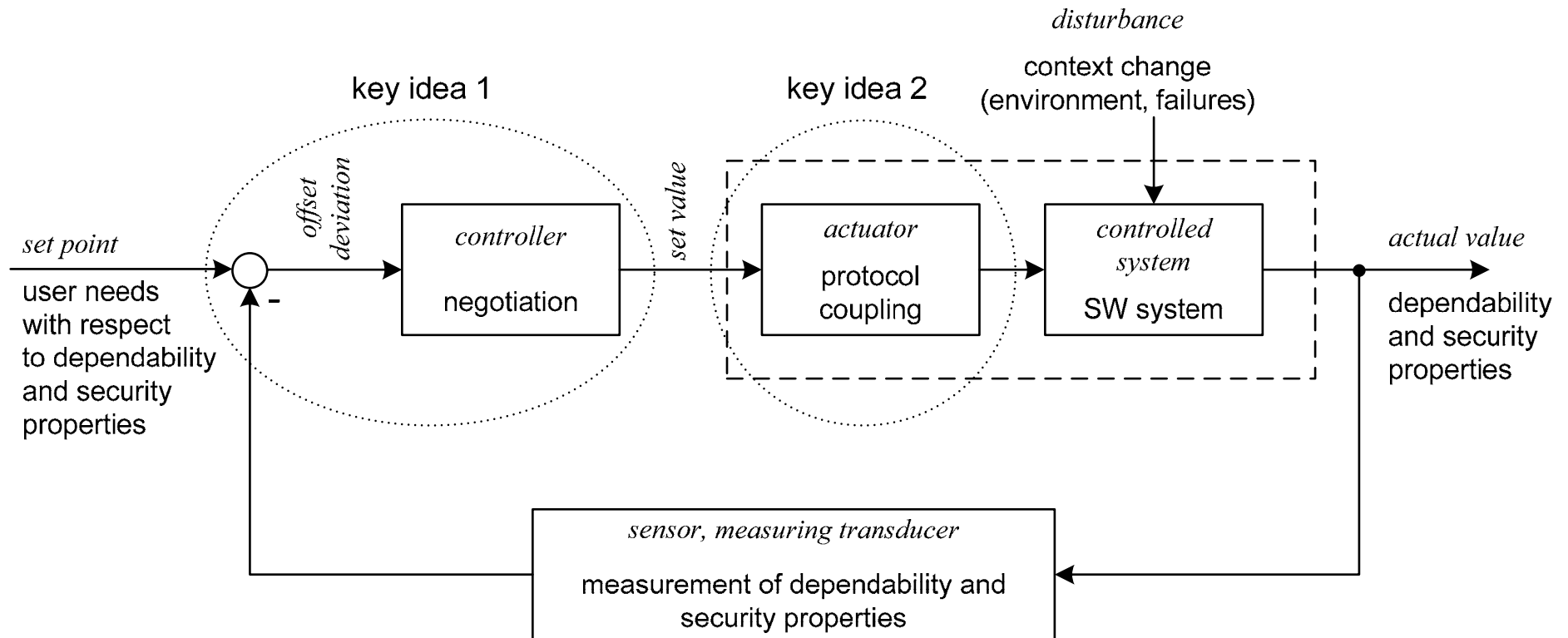
- ❑ Interceptors
- ❑ Separation of concern: AOSD
(modularization of cross-cutting concerns)
- ❑ Computational reflection:
run-time modification
- ❑ Component-based design:
Dynamic (re-)composition
- ❑ MW-Application interaction: plug-in, upcall, ...
- ❑ Autonomic computing, self-*

Interceptors



Using interceptors to handle remote-object invocations.

Control loop



Lecture 2:

Communication (Part 1)

- Networking Principles
- Internet: Ethernet and TCP/IP
- Middleware and adaptivity
- Remote procedure call

Remote procedure call

- Explicit message exchange does not conceal communication (no transparency)
- Retain familiar procedural programming model (idea from 1984)
- Rely on system to bind calling and called procedure
- Use exception handling to deal with failures
- How to implement?
- How to use?

Conventional Procedure Call (2)

- Parameter passing:
 - Call by value
 - Call by reference
 - Call by copy/restore
- Library call of system calls

→ Goal: make the RPC look like a local procedure call (transparent)!

RPC implementation issues

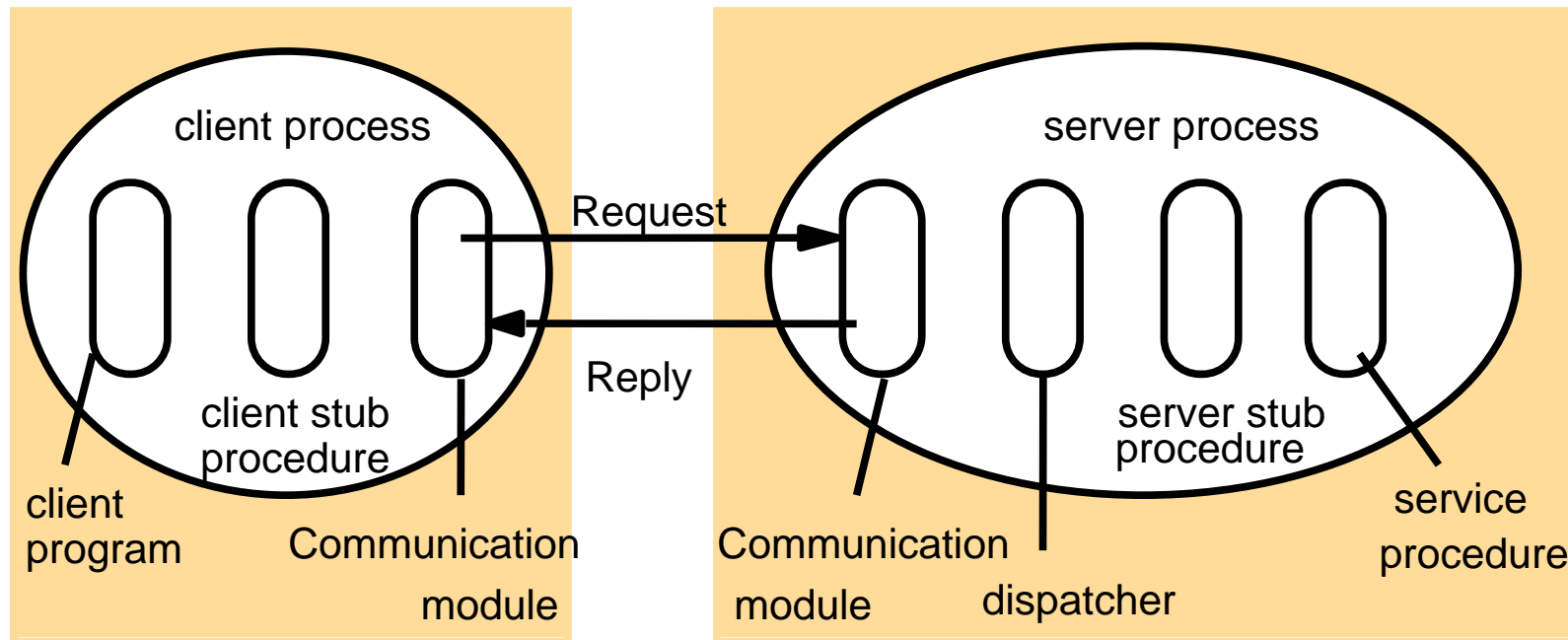
Client

- Program: whom to call?
- Kernel: what to do with data parameters?
- Kernel: where is server machine?
- Kernel: what to do with client program?

Server

- Kernel: which procedure to invoke?
- Kernel: how to pass parameters?
- Kernel: what to do with result parameters?

Runtime structure of RPC



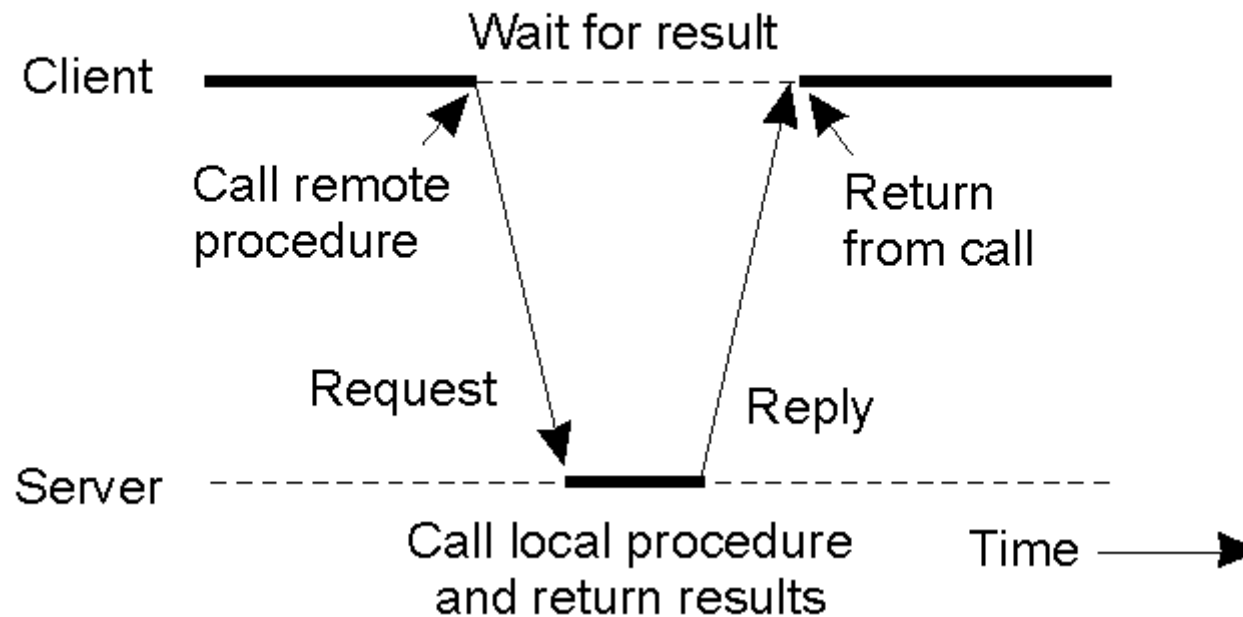
Stub procedures *marshal* and *unmarshal* parameters.

Client stub marshals input parameters (and unmarshals return parameter).

Server stub unmarshals input parameters and marshals return parameter.

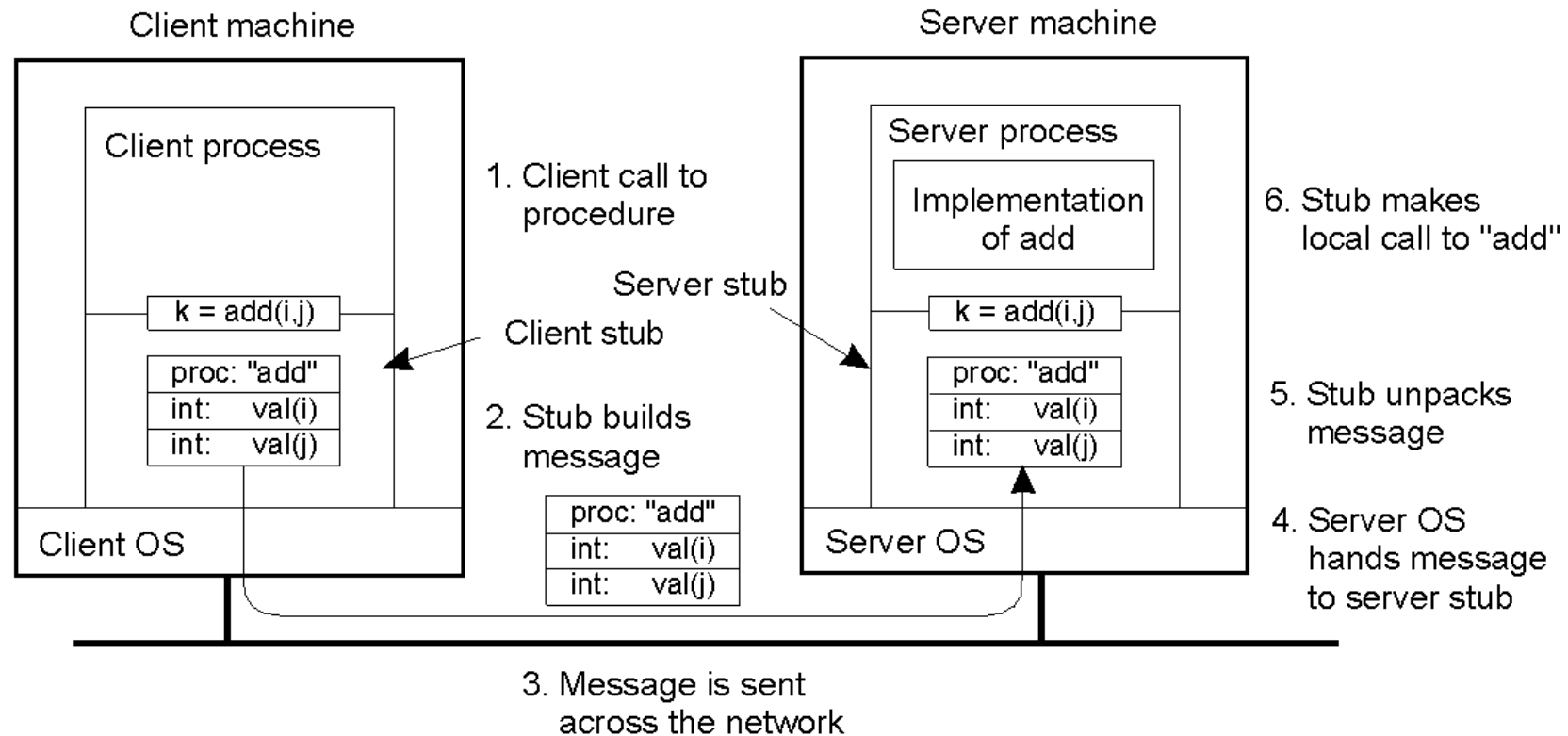
Client and Server Stubs

Principle of RPC between a client and server program.



Passing Parameters (1)

parameter marshaling



Steps involved in doing remote computation through RPC

Passing Parameters (2)

	3	2	1	0
0	0	0	5	
7	6	5	4	
L	L	I	J	

(a)

0	1	2	3	
5	0	0	0	
4	5	6	7	
J	I	L	L	

(b)

0	1	2	3	
0	0	0	5	
4	5	6	7	
L	L	I	J	

(c)

- a) Original message on the Pentium
- b) The message after receipt on the SPARC
- c) The message after being inverted. The little numbers in boxes indicate the address of each byte

Passing Parameters (3)

- Different machines and data types:
 - Character encoding (ASCII, unicode)
 - Integer format (one's/two's complement)
 - Float, double formats
 - Big/little endian (byte order/numbering)
- Passing reference parameters
 - Local address useless on different machine
 - Replace by copy/restore
 - Optimize (eliminate one copy if applicable)
 - OR: pass pointer and generate call-back function to reference arbitrary data structure (but relaxes transparency)

Parameter Specification, Stub Generation

- Caller and callee have to agree on:
 - Message format
 - Data representation
 - Message exchange (protocol)

```
foobar( char x; float y; int z[5] )  
{  
  ....  
}
```

(a)

(a) A procedure.

foobar's local variables	
	x
y	
5	
z[0]	
z[1]	
z[2]	
z[3]	
z[4]	

(b)

(b) The corresponding message.

Interface Definition Language (IDL)

- ❑ An IDL supports the definition of procedures that may be called remotely
- ❑ It provides a set of “universal” types
- ❑ IDL compiler generates client and server header files and stubs
- ❑ Client code includes client header file and links to client stub
- ❑ Server code includes server header and links to server stub
- ❑ Caller and callee need not be in same language!

Summary

- ❑ Circuit vs. packet switched and Routing
- ❑ Connectionless (datagram) vs. connection-oriented (virtual path)
- ❑ ISO OSI reference model
- ❑ Ethernet for LAN
- ❑ IP is the dominant network layer
- ❑ UDP and TCP support direct programming of communication among processes
- ❑ Middleware and flexibility
- ❑ RPC supports inter-language calls across machines. Primary drawback: Passing complex referenced parameter structures