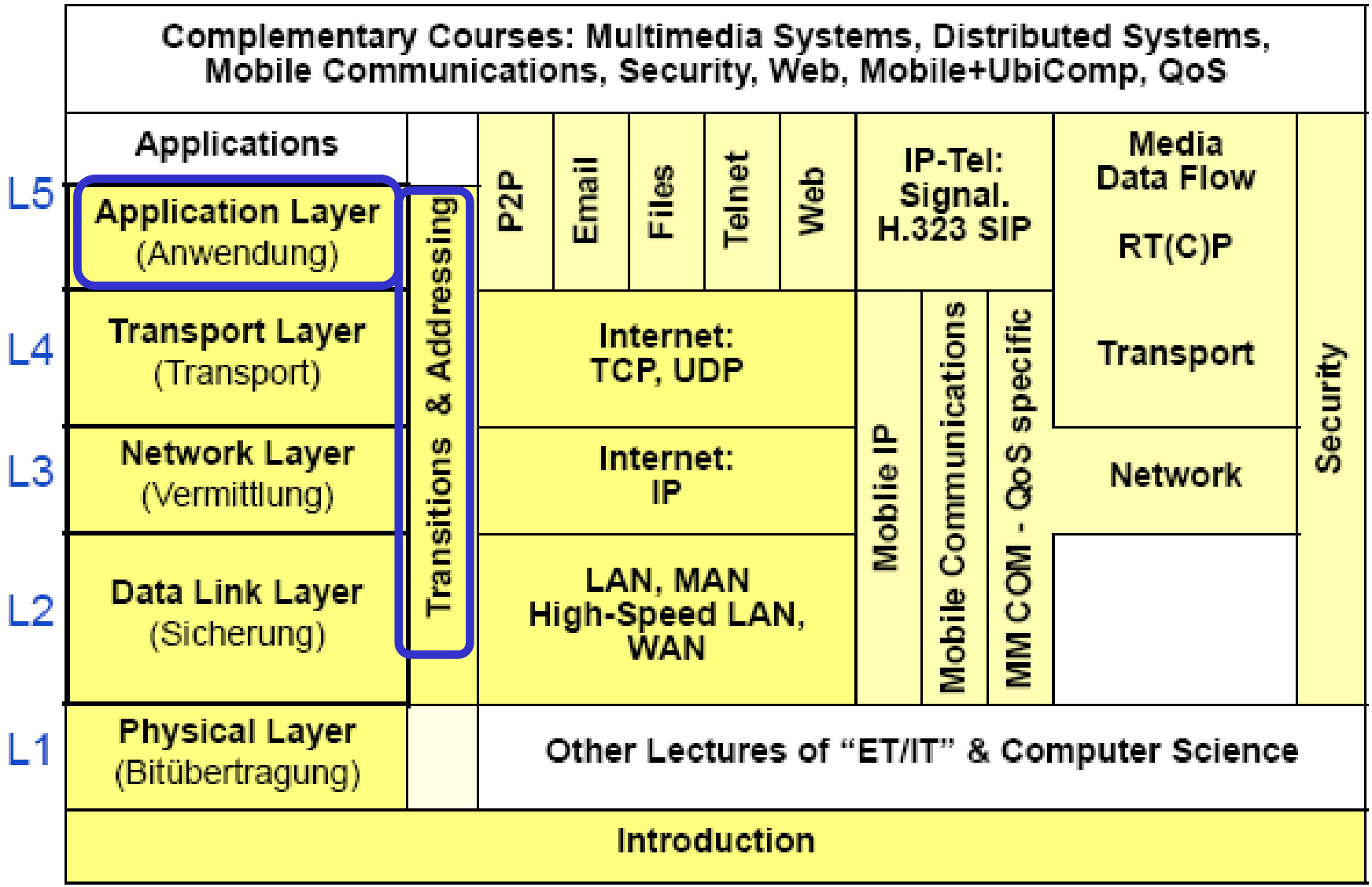
Computer Networks I

Application Layer

Prof. Dr.-Ing. **Lars Wolf**

IBR, TU Braunschweig
Mühlenpfordtstr. 23, D-38106 Braunschweig, Germany,
Email: wolf@ibr.cs.tu-bs.de

Scope


Complementary Courses: Multimedia Systems, Distributed Systems, Mobile Communications, Security, Web, Mobile+UbiComp, QoS
Applications
L5
Application Layer (Anwendung)
L4
Transport Layer (Transport)
L3
Network Layer (Vermittlung)
L2
Data Link Layer (Sicherung)
L1
Physical Layer (Bitübertragung)
Transitions & Addressing
P2P
Email
Files
Telnet
Web
IP-Tel: Signal. H.323 SIP
Media Data Flow
RT(C)P
Transport
Internet: TCP, UDP
Internet: IP
LAN, MAN High-Speed LAN, WAN
Moblie IP
Mobile Communications
MM COM - QoS specific
Network
Security
Other Lectures of "ET/IT" & Computer Science
Introduction

Overview

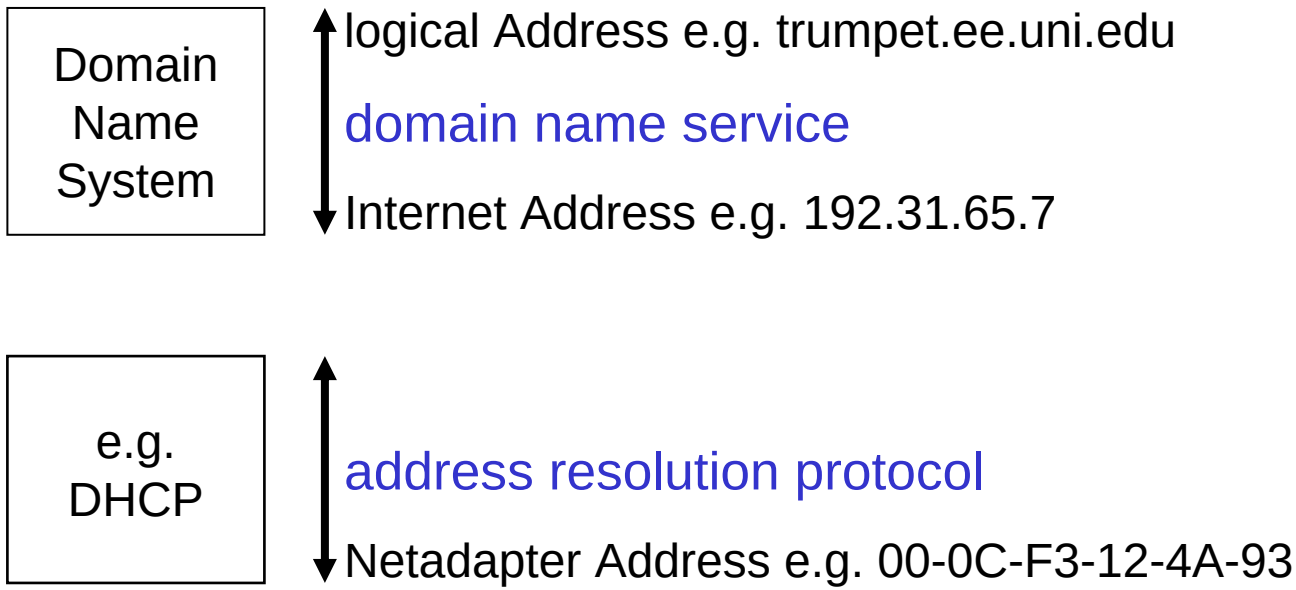
Overview

1 Addressing in General

Addressing means 3 types of identifiers: names, addresses and routes

"The NAME of a resource indicates WHAT we seek,
an ADDRESS indicates WHERE it is, and
a ROUTE tells HOW TO GET THERE [Shoch 78]

Addressing must occur at many levels of abstraction,
➔ e.g.



Addressing in General

address identifies type of or specific

- application (e.g. ssh client)
- user (e.g. in instant messageing system, e.g. in IP-telephony skype)
- service (e.g. outlook directory)
- network (e.g. subnet)
- machine (e.g. IP address, peer in P2P overlay network)
- interface (e.g. network address), ...

involves also (in general)

- overlay networks
  - in Peer-to-Peer use of distributed hash tables DHT
- directory services
- OSI, X.25 addr.
- IP addresses, incl. IP v.6
- network addr.
- Mobile IP addr.

2 Domain Name Service (DNS)

Purpose:

- Internet Protocol address is a 32-bit integer
- People prefer to assign machines pronounceable names (host names)

  - with “tv” domain of Tuvalu (Islands in South Pacific)
- hard-coded IP addresses within applications may become outdated
  - e.g., when moving mailserver / web server to other server with different address

➔ mapping from name to IP address needed

Approaches:

- use file with mapping on every host ("hosts" file), updated regularly
  - doesn’t scale nowadays (file too large, too many file update operations)
- use of decentralized hierarchical scheme

➔ DNS

Domain Name Service (DNS) - Basics

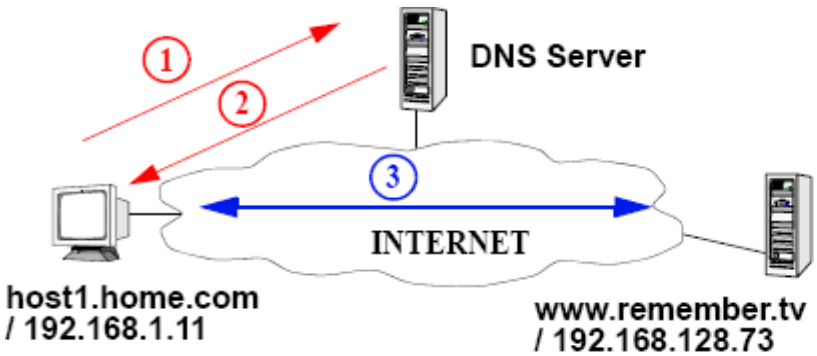
Standards:

- Basics: RFC 1034, RFC 1035
  - and lot of documents describing additional features

DNS characteristics:

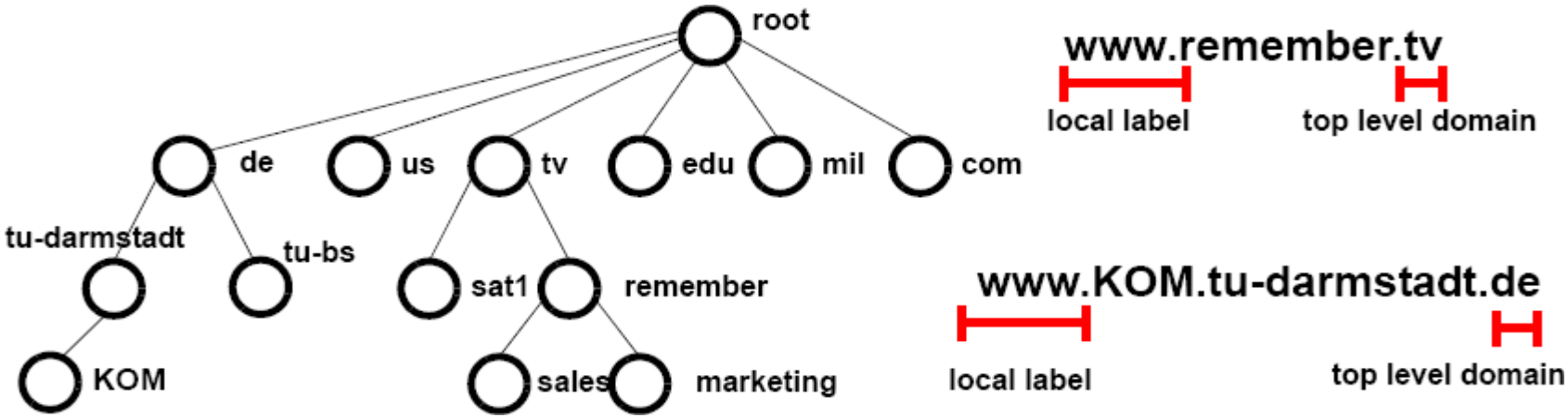
- naming scheme: centralized
- implementation scheme: distributed (responsibility and physical) database
- provides mapping between host names and IP addresses
- additional services: e.g. mail routing information

Operation - basic description (requesting a www site):



1. Host1 sends a DNS request to its DNS Server and asks for the IP address of www.remember.tv
2. The DNS server sends the IP address (192.168.128.73)
3. Host1 is now able to communicate with www.remember.tv
   tv = top-level domain of island Tuvalu

2.1 DNS: Name Space



Top-level domains
- unnamed root
- 1 arpa domain (arpa)
- generic domains:
  - most / traditionally 7*3-char. domains (com, edu, gov, int, mil, net, org)
  - now also other like .info
- country domains: based on (2-char.) country codes (ISO 3166: tv, de, ...)

Registration
- geographical (e.g. remember.tv)
- organizational (e.g. remember.com)

Domains, subdomains, ...
- by local authorities (e.g. admin of remember)
- e.g. sales, marketing, ...

DNS: Name Space

Tree leaves represent domains without further subdomains
- but with IP equipment (computers, printers, ...)

Distribution with regard to organizational issues
- but without regard to physical connections
- hierarchy can be distributed at the underlaying network

Allows multiple naming hierarchies to be embedded
- specified by object types: e.g. MX: Mail Exchanger, NS: Name Server

Several domains can be hosted by one server
- e.g. domains sales.remember.tv, marketing.remember.tv hosted by one server

'Popular' domains have been used up
- especially in .com

New top-level domains have been approved by ICANN
- The Internet Corporation for Assigned Names and Numbers
- www.icann.org

2.2 DNS: Name Server Types

No server has all name-to-IP address mappings

Local name servers:

- each ISP, company has local (default) name server
- host DNS query first goes to local name server

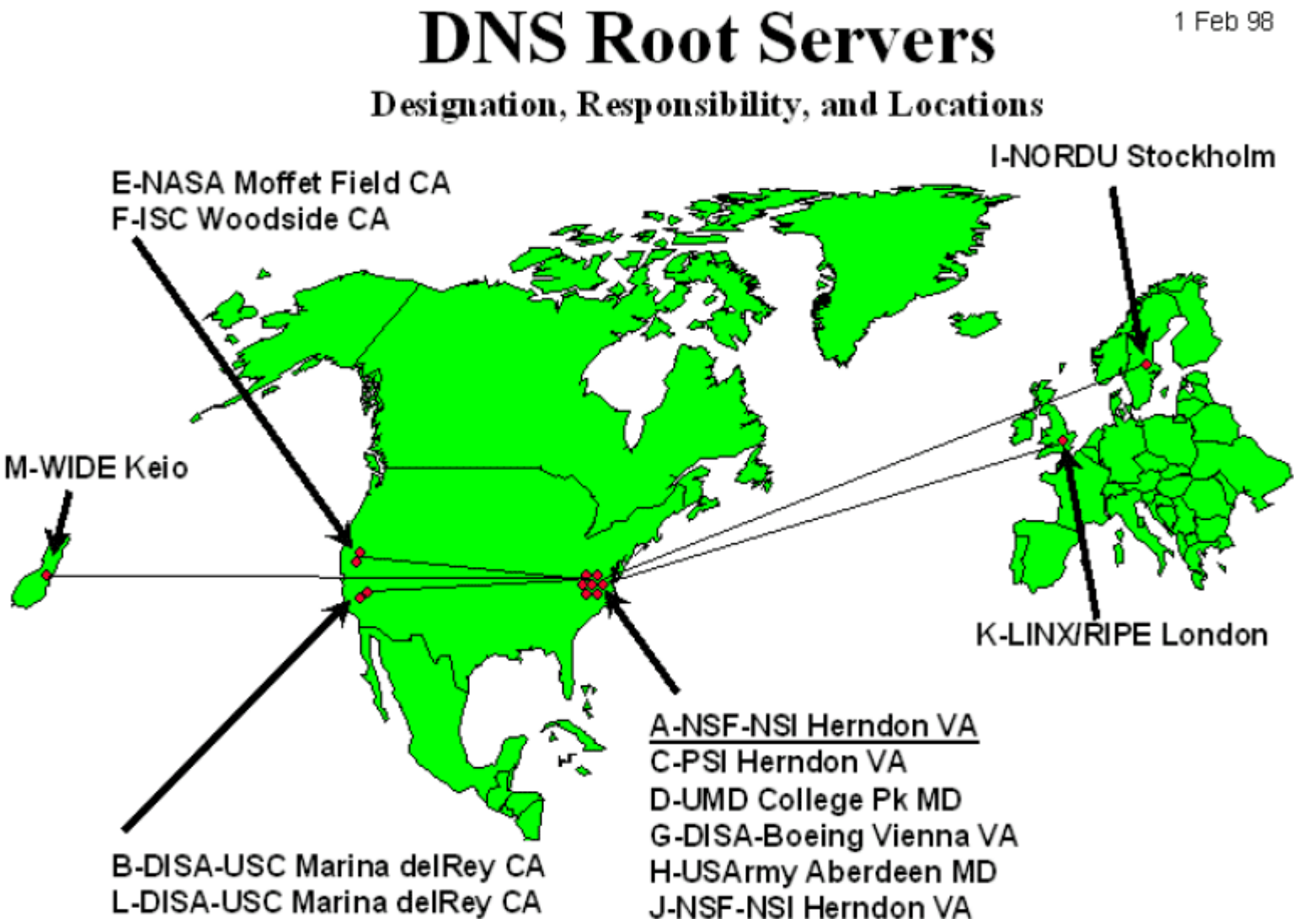
Authoritative name server:

- for a host: stores that host's IP address, name
- can perform name/address translation for that host's name

Root name server:

- contacted by local name server that can not resolve name
- root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
- returns mapping to local name server

DNS: Root Name Servers



DNS: Resource Records

Each domain can have set of RESOURCE RECORDS (RR) associated with it

- Different types: most common are IP address

DNS maps domain names onto resource records

Resource record format are five tuples:

Domain_name	Time_to_live	Class	Type	Value
domain to which this record applies	'stability' of the record	IN for Internet information (others possible)	record type (see below)	number, domain, ASCII string depends on type

DNS: Resource Records

Some record types:

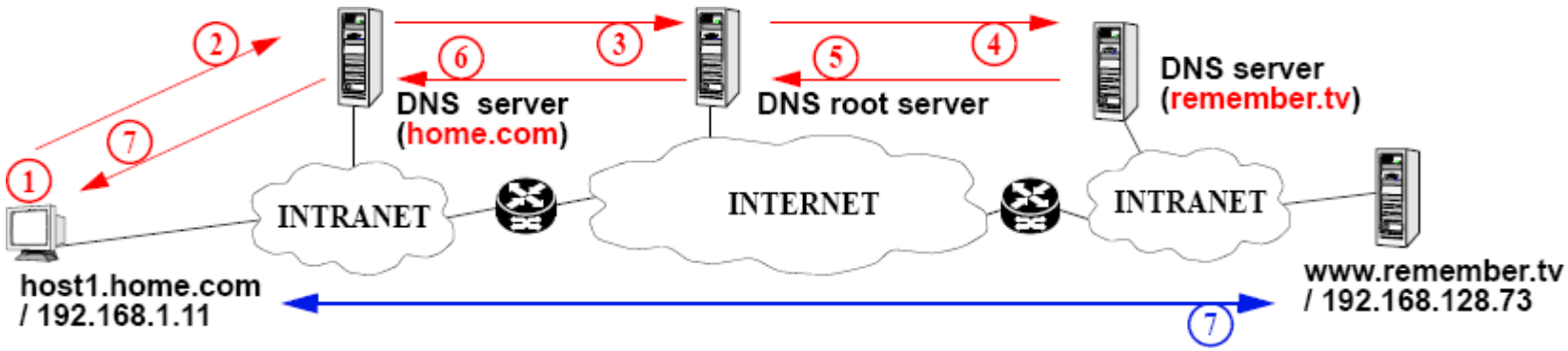
Type	Meaning	Value
A	IP address of named host	32 bit integer giving IP address
MX	Mail exchange associated with name	Priority, domain willing to acceptemail
NS	Name server	Name of server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
…		

DNS Database

Example:

```
$TTL 86400
;
; Start of Authority:
; Nameserver:
@ IN NS agitator.ibr.cs.tu-bs.de.
@ IN NS infbssys.ips.cs.tu-bs.de.
@ IN NS oker.escape.de.
;
; Mail Exchanger fuer ibr.cs.tu-bs.de:
;
@ IN MX 10 agitator.ibr.cs.tu-bs.de.
cip  IN MX 10 pott.cip.ibr.cs.tu-bs.de.
mail.cip   IN CNAME   pott.cip.ibr.cs.tu-bs.de.
;
; IPv6:
;
ipv6 IN NS agitator.ibr.cs.tu-bs.de.
IN      NS      oker.escape.de.
IN      NS      ns.ipv6.tm.uka.de.
asaft   IN A  134.169.34.100
IN   MX 10 agitator.ibr.cs.tu-bs.de.
osaft   IN A  134.169.34.101
IN   MX 10 agitator.ibr.cs.tu-bs.de.
salvator   IN A       134.169.34.17
IN   MX 10 agitator.ibr.cs.tu-bs.de.
nis  IN CNAME salvator
loghost    IN CNAME   salvator
```

2.3 DNS: Protocol



typical operation - extended description

DNS recursive resolution:

- in many steps
  1. local appplication wants to resolve address
  2. Host1 sends a DNS request to its local DNS server and Host1 asks for the IP address of www.remember.tv...
  3. ...
  4. ...
  5. ...
  6. ...
  7. Host1 is now able to communicate with www.remember.tv

DNS: Protocol

1. Application on Host1
   - calls local “resolver”, asks for IP addr. of www.remember.tv (name as parameter)
2. Host1
   - sends a DNS request (using UDP) to its local DNS server and asks for IP address.
3. DNS server can not resolve the request
   - forwards the request to one of the toplevel root server
   - request marked as “recursive resolution”
4. toplevel DNS server
   - knows the location of the DNS server(s) responsible for remember.tv
   - request is (also recursive) forwarded to this DNS server
5. DNS server
   - is capable to resolve the request
   - sends the IP address (192.168.128.73) back to the root server
6. root server
   - sends the answer to the home.com DNS server

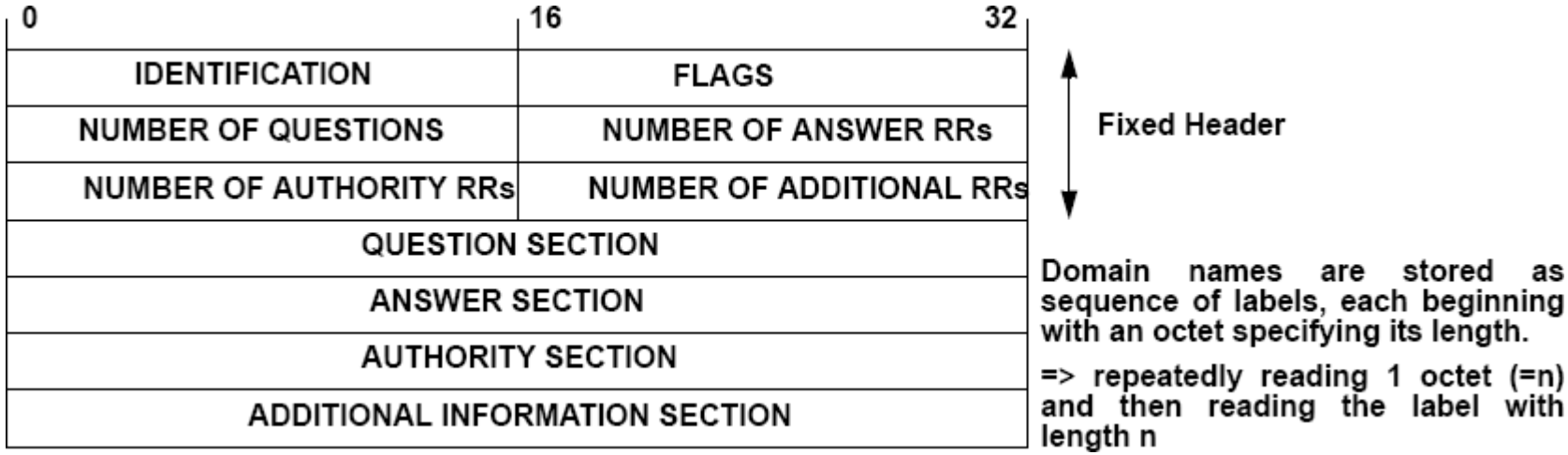
DNS: Protocol

7. home.com DNS server
   - sends the answer to host1
8. Host1 is now able to communicate with www.remember.tv

➔ Obviously optimizations are necessary
➔ Efficient Translation, Caching name server

DNS: Protocol

Domain Server Message Format:



Client fills in only the QUESTION SECTION,
server returns Questions and Answers in response
The QUESTION SECTION contains the queries consisting of:

- QUERY DOMAIN NAME: the name (stored as labels)
- TYPE: e.g. MX Mail Exchanger, NS Name Server
- QUERY CLASS: different classes, e.g. official Internet names

The ANSWER SECTION contains the answers consisting of:

- TIME TO LIVE: specifies how long the information can be cached
- RESOURCE DATA: each record describes one domain name and mapping
- Other fields: TYPE, CLASS, RESOURCE DATA LENGTH, RESOURCE

2.4 DNS - Summary

Transport protocol:

- normally UDP (PORT = 53) due to efficiency
- TCP also possible (due to security reasons)

Host names <-> IP addresses

- 1 host : 1 IP addr
- n hosts : 1 IP addr
  - many host names may correspond to a single IP address
  - facilitates fault tolerance and load distribution
  - allows a site to move physical location without being noticed at DNS level
- 1 host : n IP addr
  - a single host name may correspond to many IP addresses
  - allows a single machine to be addressed via many network interfaces

e.g. web site //remote.12dt.com/rns/ allows for reverse look-up

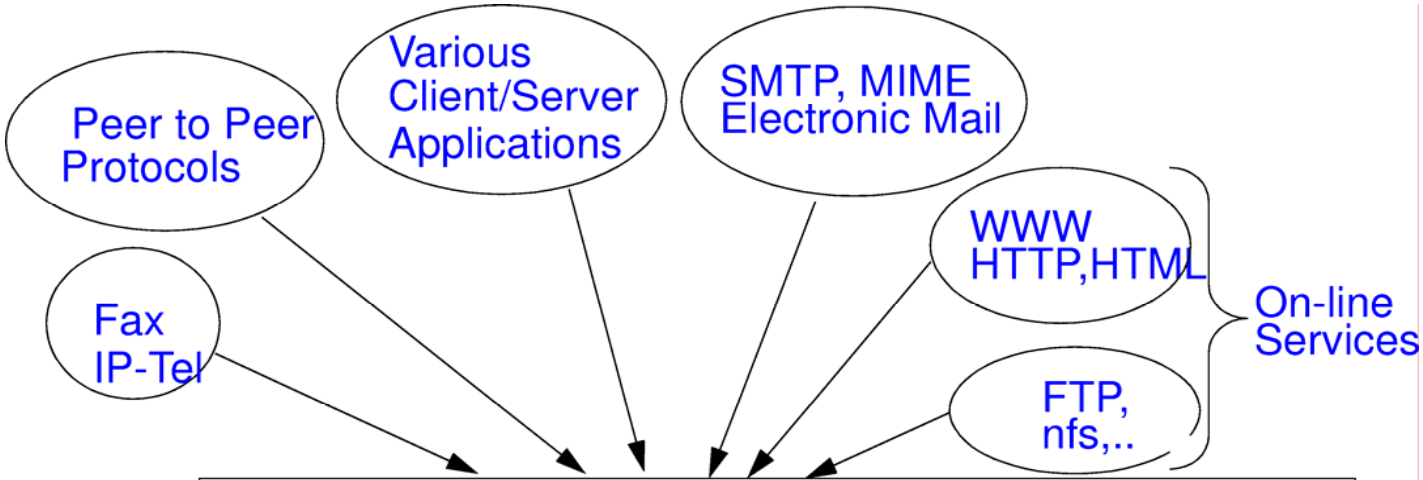
- created by Frank Riherd
- return the name of the resolved IP address

DNS

Problems:

- e.g. coexistence with DHCP
  - dynamic assignment of IP addresses
  - it is possible to integrate DHCP and DNS mechanisms
- e.g. security
  - DNS system is often used for attacks
  - e.g.
    - an attacker modifies the mapping by spoofing packets
    - Countermeasures:
      TCP transport protocol
      DNS authentication mechanisms

1 Application-Oriented Communication Services



L5	**Application-oriented services**
L4	**Transport layer**
L3	**Network layer**
L2	**Data link layer**
L1	**Physical layer**

2 Session and Application Layer

Approaches for developing distributed programs

1. COMMUNICATION ORIENTED approach
   - to define messages and formats
   - to use e.g.
     - client-server paradigm
       - defined as reactions to incoming messages
     - sockets

→ evaluation:
   - benefits
     - all communication is handled similarly
   - disadvantages
     - program design depends on type of communication
     - an error in a protocol may lead to complete redesign of the program
     - development of communication protocols may be complex

2. APPLICATION ORIENTED approach
   - to use programming paradigm
   - to transfer modularization approach and object orientation to distributed programming
   - functionality located in procedures/objects
     - not in communications
   - communications between systems is independent of programs
     - e.g. by using the Remote Procedure Call concept

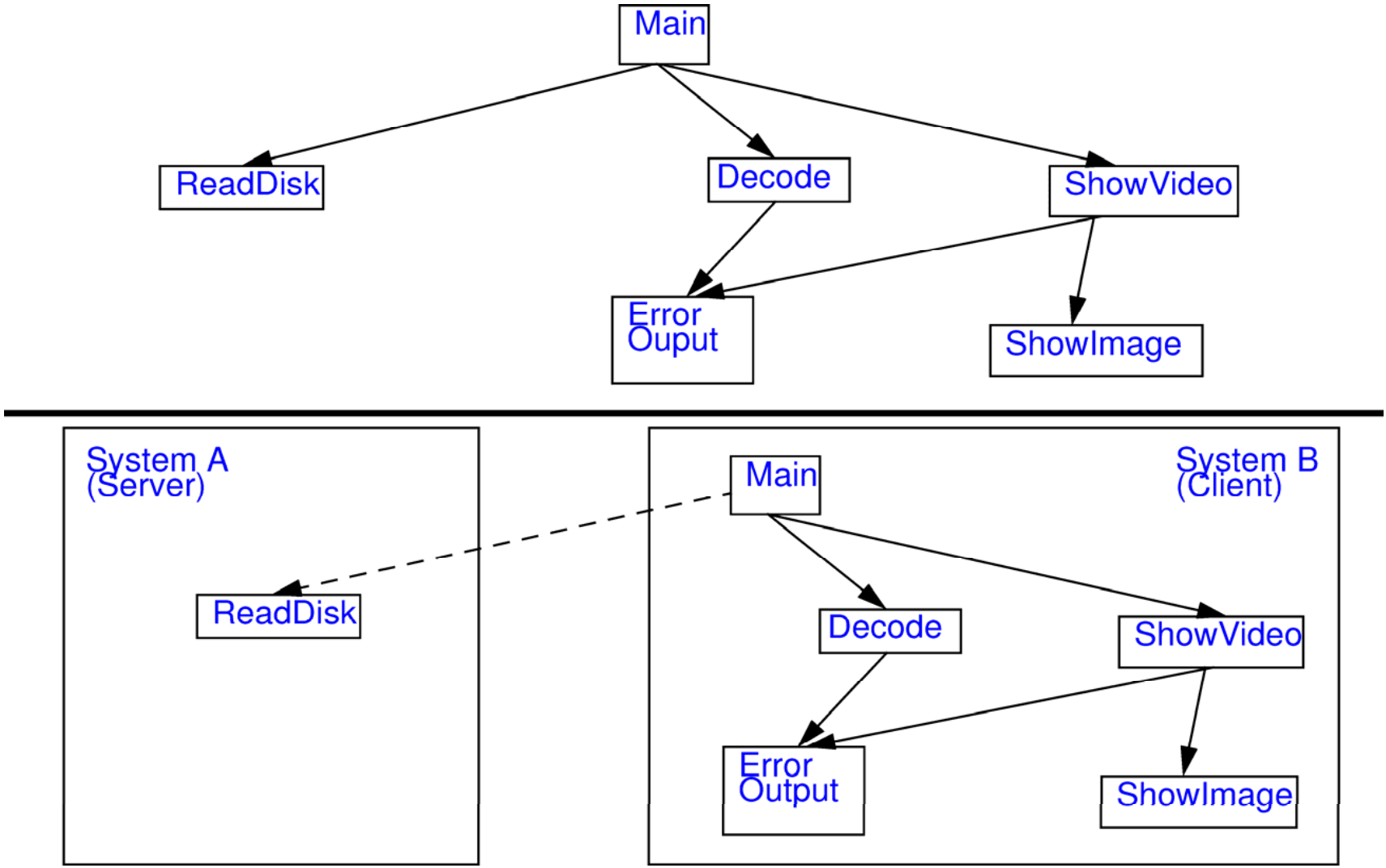
Session

Tasks
- to control the dialog (interaction) between peers
- to synchronize the data transfer

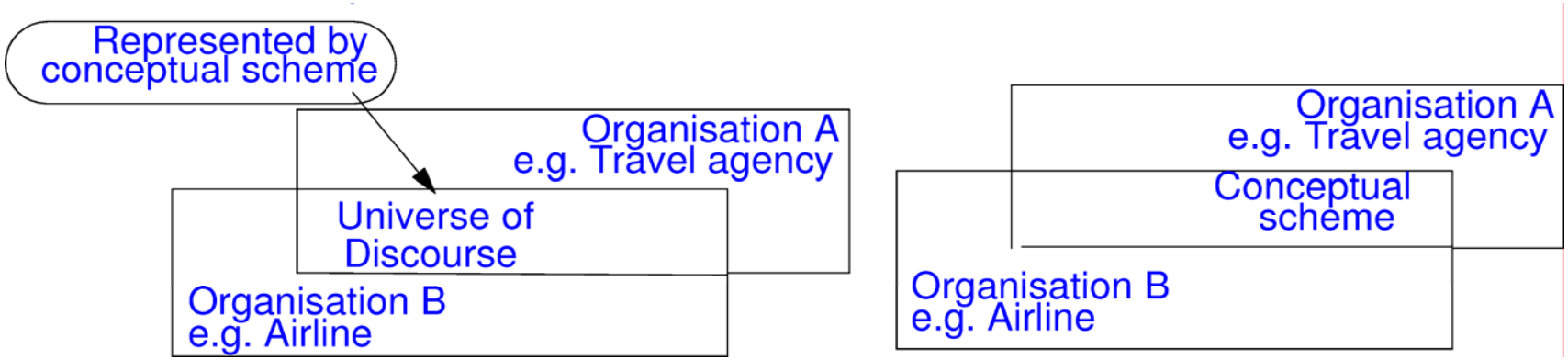
Mechanisms
- Tokens
  - to initialize services under comtrol of a user
  - E.g.
    - data token, release token, synchronize-minor-token, major/activity token
- Synchronization
  - To monitor communications by insertion of synchronization markers
    - (i.e. “Wiederaufsetzungspunkte”)
  - To resynchronize
    - To reset communications to a well defined previous state
- Activity
  - A session may comprise many activities
  - An activity can be interrupted and later on resumed
  - A session keeps track of the related activities

Parameters
- Security of a session
- Priority of a session
- Throughput
- ..

Session: Example



3 Data Presentation



Sender and receiver need common data presentation to allow understanding

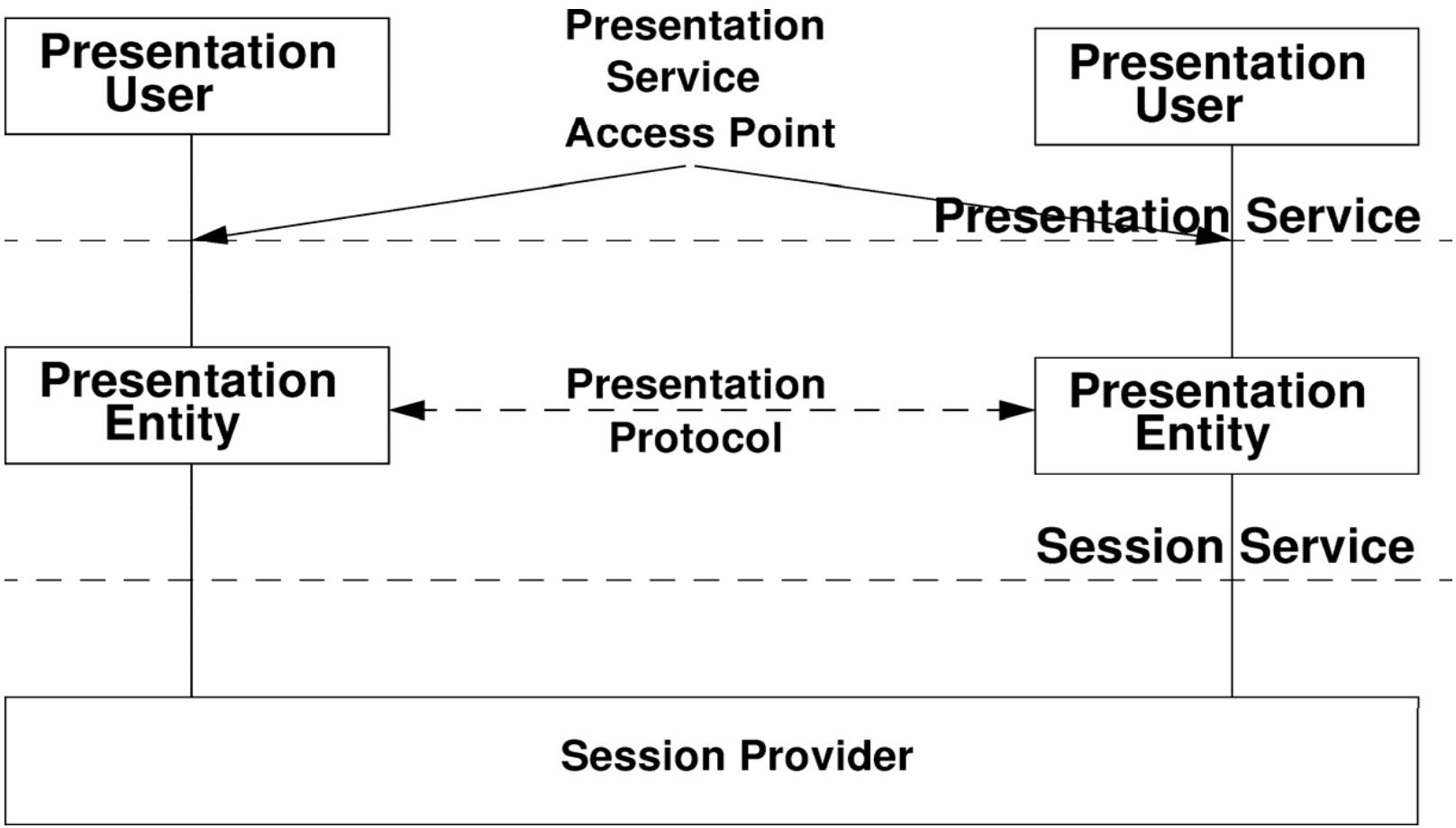
- 'communication' of content
  - not of bits
- needed for formats, data types, compression, coding, ...

Generic view:

- Universe of Discourse
  - part of the real world which is to be processed in the system
- Requirements
  - relation to the same universe of discourse
  - common conceptual scheme
  - comprehensible representation of the conceptual scheme's objects (i.e. data conversion) to both communication parties
- Conceptual scheme
  - formal description of the universe of discourse

3.1 Task

To provide well understood data presentation for any communication between open systems bb2



bb2 deleted plural 's' from communication

blume; 30.05.2006

Task

Functions:

- to transfer communication control services
- to allow the specification of complex data structures
- negotiation of required data structures
- to convert the local representation into a global one

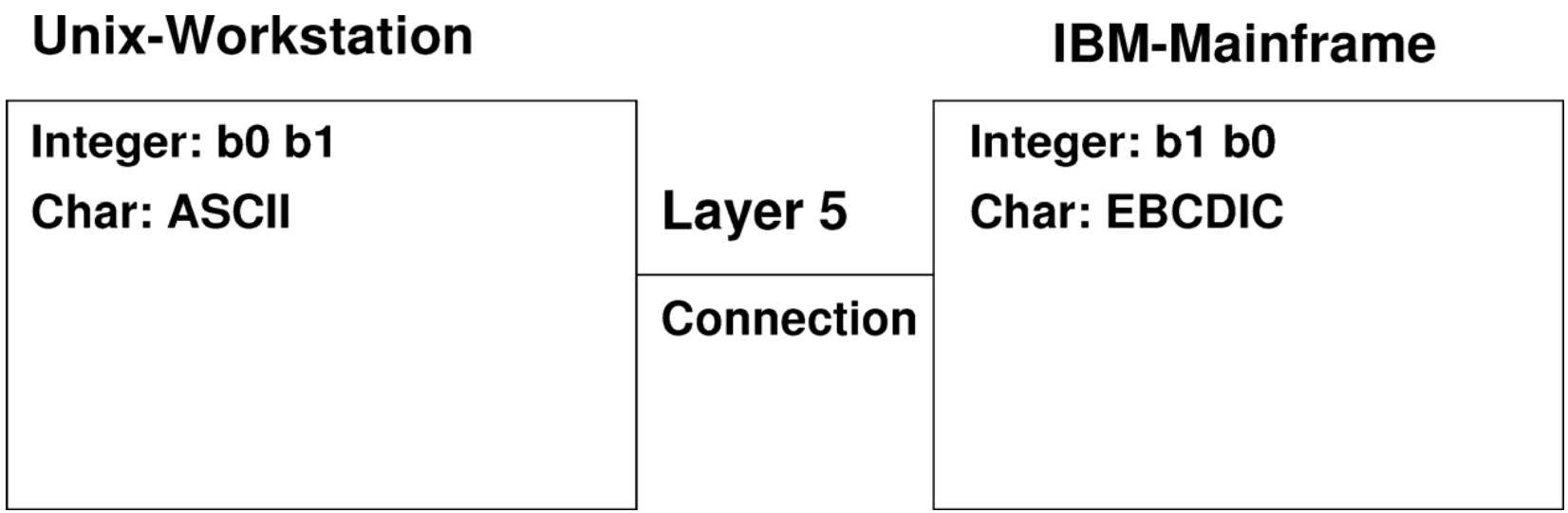
Because

- connection does not mean communication
- communication implies a common understanding

Example:

- understanding the words
  - Igel (German) - eagle (English)

3.2 Example



Situation:

- even though correct communication at lower layers there is no further communication possible

→ Semantics are lost

- heterogeneous software

→ Coding regulations depend on compiler

- distributed objects
- E.g. Common Object Request Broker Architecture CORBA

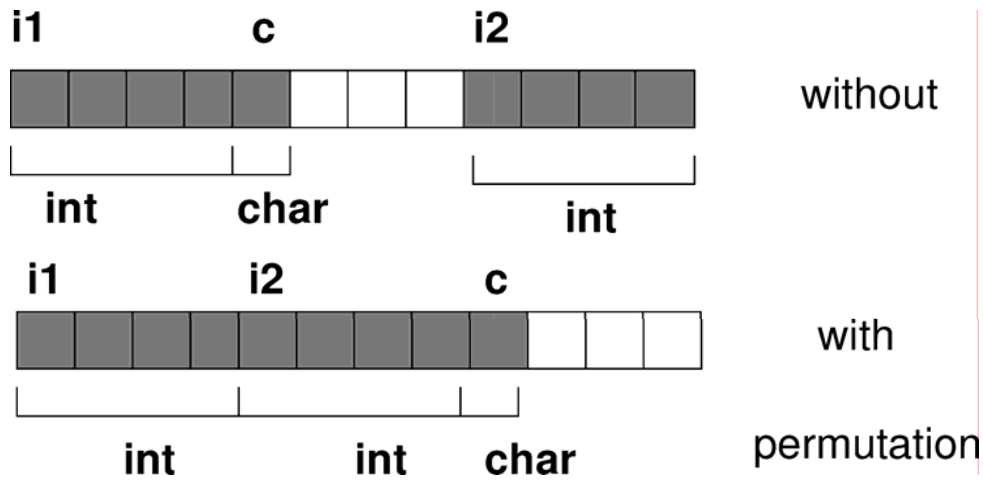
Example

Example:

```
struct {
    int     i1;
    char    c;
    int     i2;
}
```

- char: one byte, no alignment conditions
- int: 4 bytes, alignment according to an address divisible by 4

e.g. see compilation with and without permutation strategy:



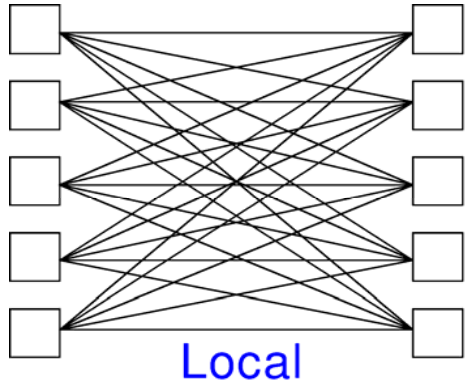
Coding

Type (c)	Coding Rules
INT	Length Coding type Arrangement Justification (with word border)
FLOAT	Length of mantissa Length of the exponential Exponential basis Coding type Arrangement Justification
CHAR	Coding type

3.3 Data Presentation: Methods

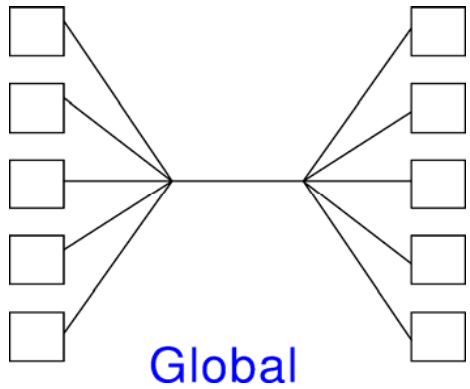
Local presentation of a communication partner

- n x (n-1) conversion routines
- a maximum of one conversion per relationship
  - local format f1 directly to local format f2



Global presentation

- 2 x n conversion routines
- 2 conversions per relationship
  - local f1 to global g,
  - global g to local f2
- standards:
  - XDR
  - ASN.1
- scheme:



3.4 XDR: External Data Representation

XDR: External Data Representation
- presentation layer with very low functionality

BIG-ENDIAN
- byte 0 as the most significant (i.e. left)

versus
LITTLE-ENDIAN
- byte 0 as the least significant (i.e. right)

all data is mapped to a pre-defined transfer syntax
- (no negotiations)
- all integers as 4-byte big-endians
- floating-point numbers in IEEE format:
  - mantissa 23 bits
  - exponential 8 bits
  - algebraic sign 1 bit
- texts in ASCII code
- all data elements aligned with 4-byte limit

Disadvantage:
- Two systems with completely identical representations have to convert twice

bb3

bb3 deleted 'which are', inserted 'with'

blume; 30.05.2006

Big Endian and Little Endian

most significant			less significant
byte 1	byte 2	byte 3	byte 4

Big Endian

- e.g. Motorola 68x0, IBM 370 (Big Endian)

less significant			most significant
byte 1	byte 2	byte 3	byte 4

Little Endian

- e.g. Intel

comment: usually also relates to bits

Big Endian and Little Endian

below the excerpt from a respective configuration header =file in a UNIX system

- /* Definitions for byte order, */
- /* according to byte significance from low address to high. */

- #define LITTLE_ENDIAN 1234
  /* least-significant byte first (vax) */

- #define BIG_ENDIAN 4321
  /* most-significant byte first (IBM, net) */

- #define PDP_ENDIAN 3412
  /* LSB first in word, MSW first in long (pdp) */

- #define BYTE_ORDER BIG_ENDIAN

XDR, the Representation „Layer“ of the Internet

Essential component: XDR compiler

- generates
  - C data structures compatible with the XDR definition and
  - program pieces for coding and decoding

Summary/example of a typical data packet

start of the packet
Ethernet header
IP header
UDP header
RPC header
User data in XDR format
Ethernet checksum
end of the packet

Comment: XDR does NOT need its own header

Server

- provides services
- i.e.
  - waits for incoming service requests from clients
  - processes requests and sends results as response
- may use other servers to process request
  - becomes client in that case

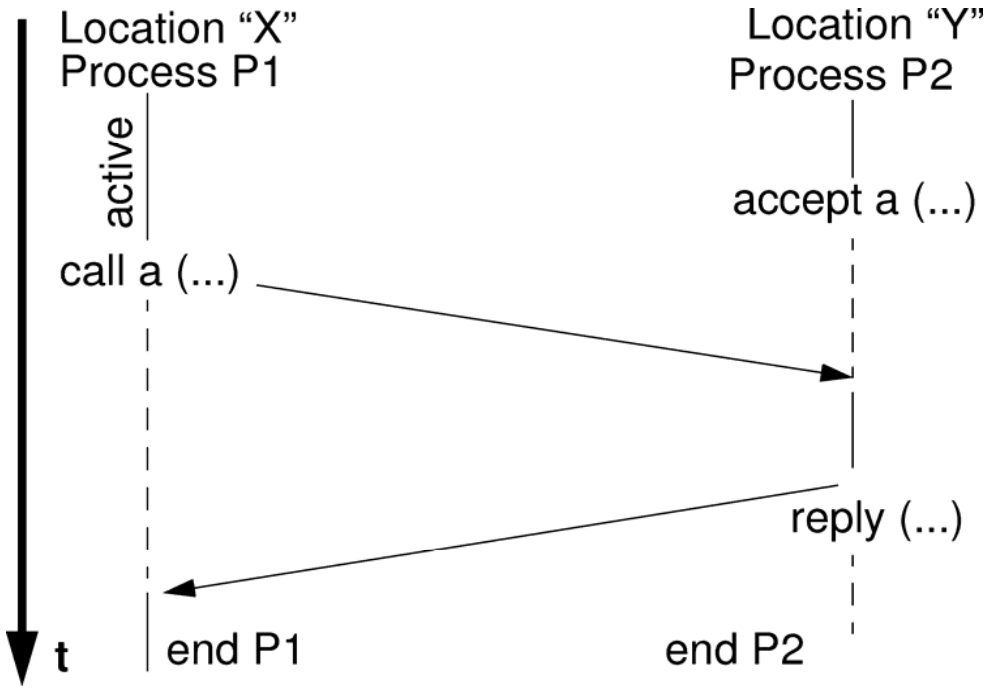
Client

- uses services provided by server
- i.e.
  - sends requests to server
  - (typically) waits for response from server

For client conceptually similar to PROCEDURE CALL

- call procedure
- wait for result

4.1 Remote Procedure Call - RPC synchronous Remote Service Invocation (sRSI)



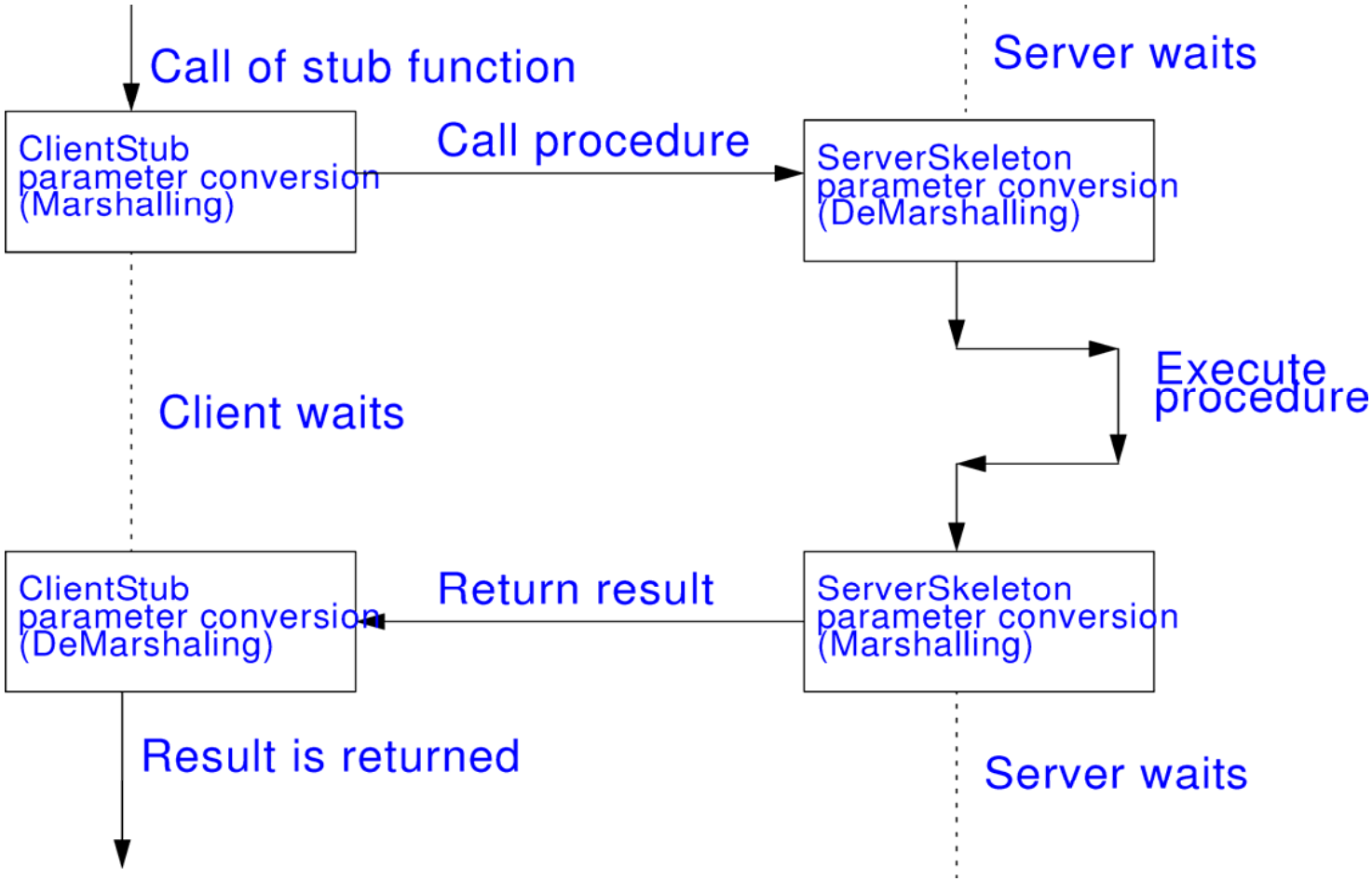
Concept

- synchronization between client and server
  - synchronous Remote Service Invocation (sRSI)
- characteristic: e.g. limited parallelism

Basic idea:

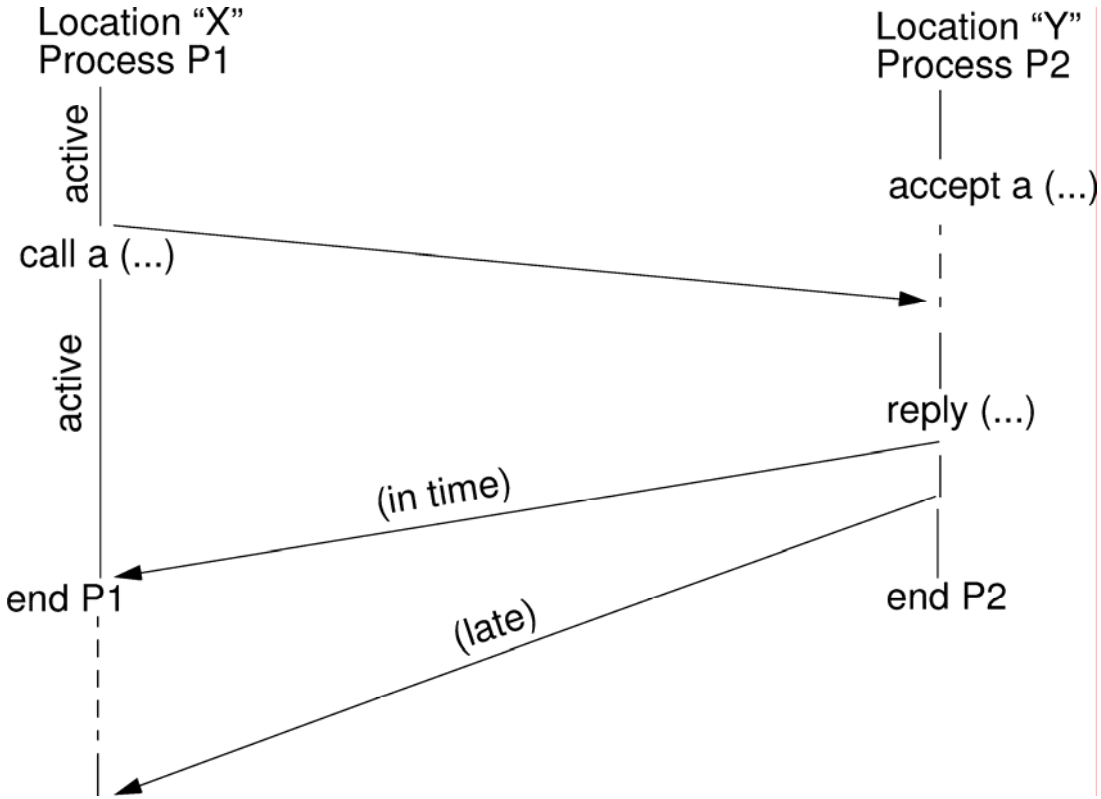
- application cannot differentiate between
  - remote procedure call and
  - local procedure call

RPC Example


Call of stub function
Server waits
ClientStub
parameter conversion
(Marshalling)
Call procedure
ServerSkeleton
parameter conversion
(DeMarshalling)
Execute
procedure
Client waits
ClientStub
parameter conversion
(DeMarshaling)
Return result
ServerSkeleton
parameter conversion
(Marshalling)
Result is returned
Server waits

4.2 Asynchronous Remote Service Invocation (aRSI)

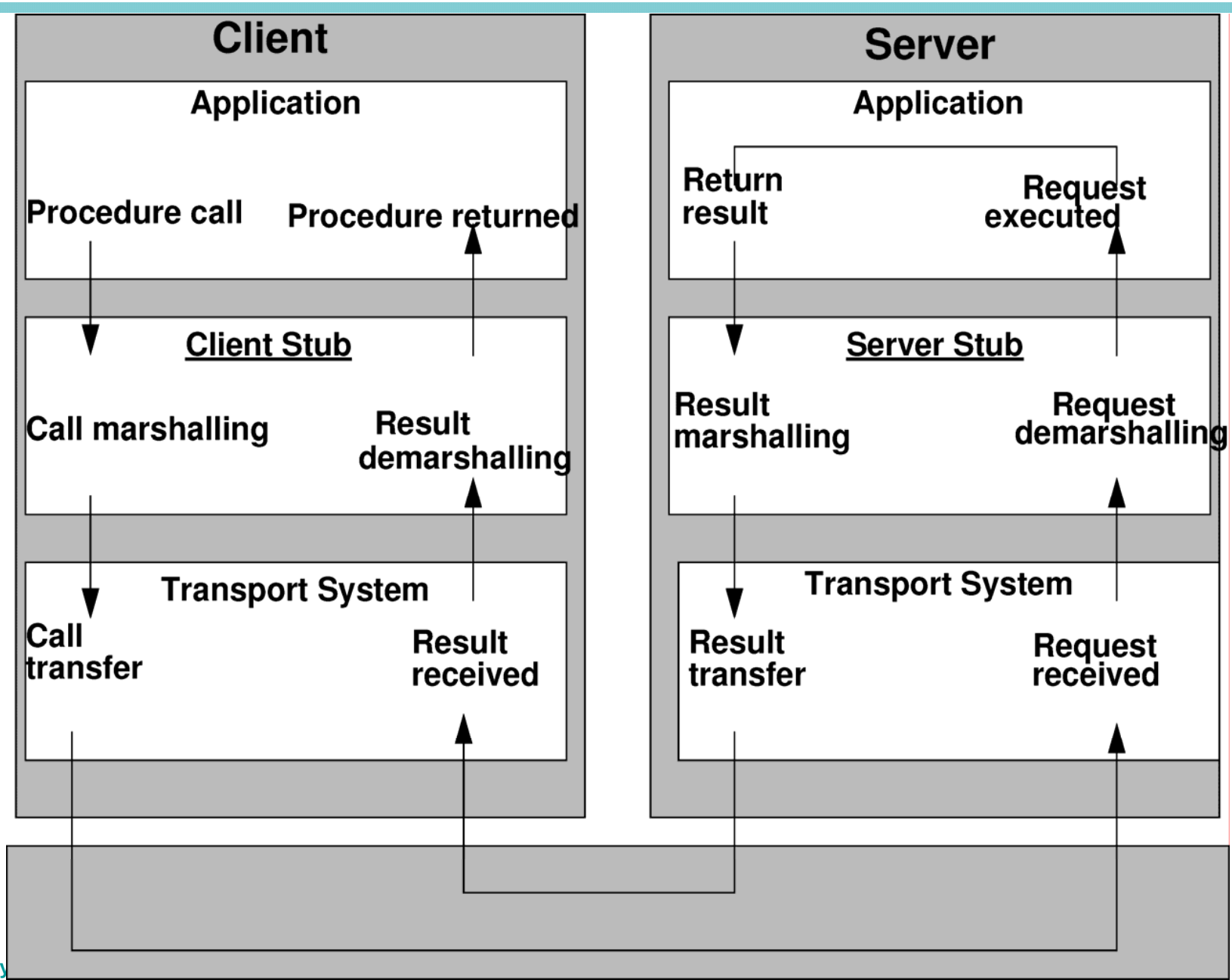
Alternatively to RPC



Characteristics (among others)

- parallelism between client and server possible
- associating requests and respective results more difficult

4.3 RPC: Cycle


Client
Application
Procedure call
Procedure returned
Client Stub
Call marshalling
Result demarshalling
Transport System
Call transfer
Result received
Server
Application
Return result
Request executed
Server Stub
Result marshalling
Request demarshalling
Transport System
Result transfer
Request received

RPC Cycle: Tasks of “Stub” Procedures

to locate and bind server with/and client
- server
  - registers its service at database (server) by providing its name (ASCII), network address and service number (any 32 bit number) (export)
- client-stub
  - sends request to database
    - its name (which is also the name of the server) in ASCII
- database service
  - returns network address and unique server identification (binding)

marshalling/demarshalling (parameter arrangement) of parameters and results (guarantees transparency)
- client
  - collects all parameters of an RPC call and
  - packs them into a message
- server
  - unpacks the parameters,
  - performs function(s) and
  - packs results into a message
- client
  - unpacks results

error treatment, error semantics

communications
- transport system interface
- data representation
- authentication/encryption

4.4 RPC: Error Semantics

Various errors may occur, e.g.

- requests or replies get lost or are garbled during data transfer
- client or server crashes while RPC is ongoing

Several error classes can be distinguished

Maybe-semantics

- server process may have been executed once

At-least-once-semantics

- server process is executed error-free at least once (if not more)

At-most-once-semantics

- server process is executed error-free at most once

Exactly-once-semantics

- server process is executed error-free exactly once (guaranteed) including transmission

4.5 RPC: Idea and Reality

Basic idea:

- application cannot differentiate between
  - remote procedure call and
  - local procedure call

Problems:

- transparency:
  - parameter treatment (“call by reference”, “pointer”, procedures ...)
  - side effects
- efficiency
  - additional effort for “marshalling/demarshalling”
  - error treatment, for e.g. recovery after a “server crash”
- conception
  - client and server roles may change
    - e.g. in streaming

Implementations

- e.g. SUN RPC (RFC 1057)
- e.g., RPC at Open Software Foundation’s (OSF) Distributed Computing Environment (DCE)

5 Middleware – Objects - CORBA

Common Object Request Broker Architecture - CORBA

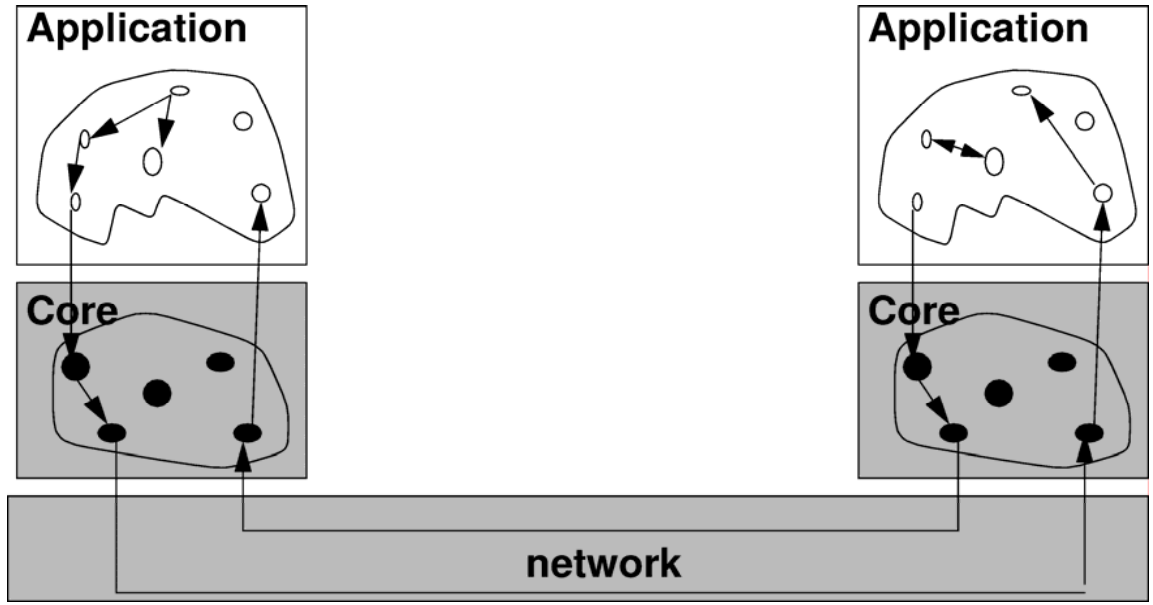
Middleware is

- software/abstraction glue which allows separate applications to communicate TRANSPARENTLY
- i.e. to inter-operate independently from
  - hardware and system devices,
  - operating systems capabilities
  - communications infrastructure

History

- 1987 - Sun RPC
- 1988 - Distributed Computing Environment (DCE) of Open Software Foundation (OSF)
  - coined term “middleware”
  - incl. naming service, fault semantics, Interface Definition Language (IDL)
  - but: no object oriented model with inheritance, static binding of declared procedure, ..
- today - Object oriented approach of Object Management Group (OMG)
  - Object Management Architecture (OMA)
  - Common Object Request Broker Architecture (CORBA)

Object Oriented Software Development



Goals

- functionality, efficiency, robustness,
- reuse, future enhancements possible

Alternative (unfortunately too often used) development methods

- “to develop from scratch”
- “Copy, Paste and Adapt individual code”
- “Combine generic parts taken from libraries”
- “Use objects, inherit from and instantiate framework components”

6 Web Services

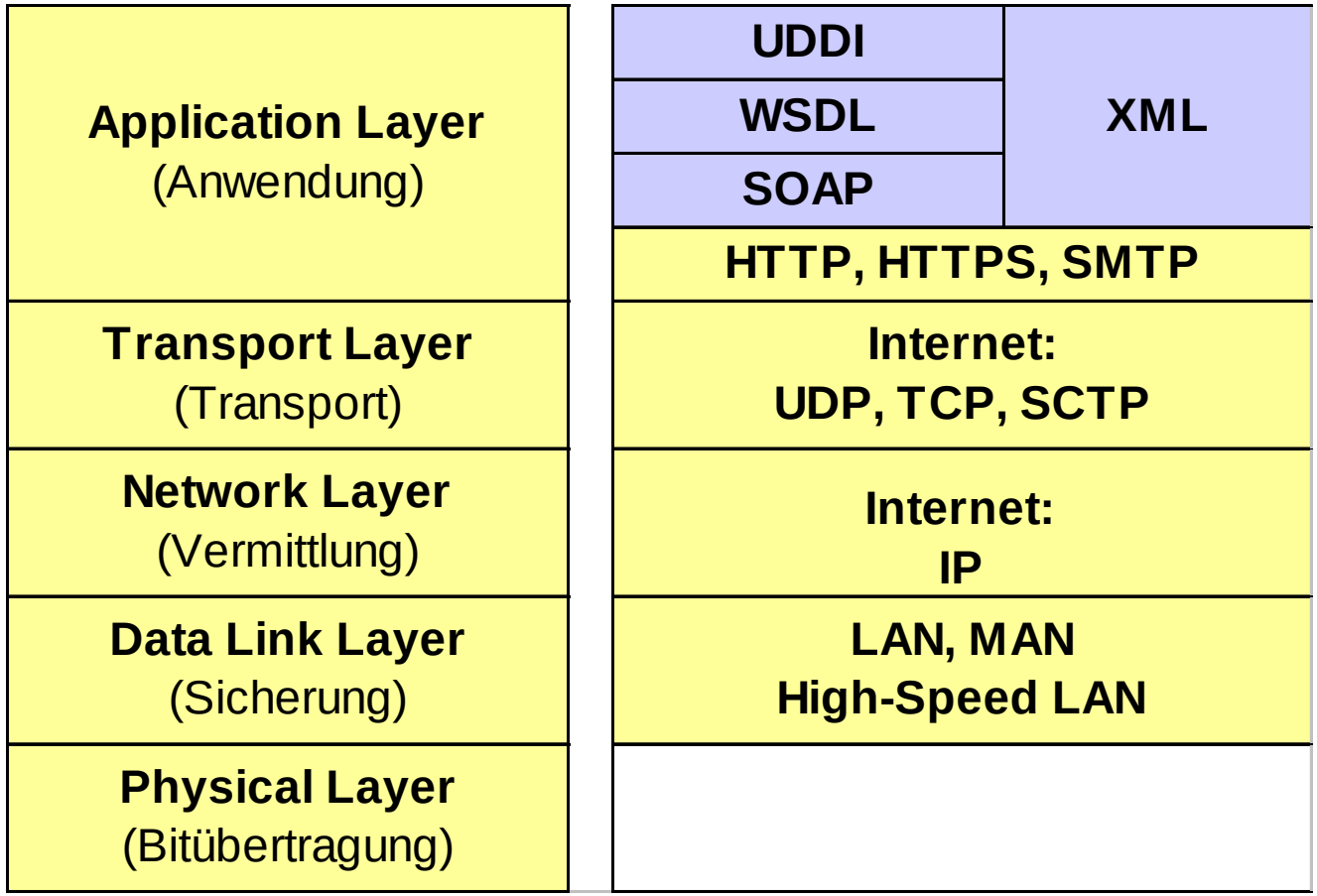
- Definition: A Web service
  - is identified by a URI,
  - has interfaces and bindings are capable of being defined, described, and discovered as XML artifacts
  - supports
    - direct interactions with other software agents using XML
      - based messages exchanged via internet-based protocols (W3C).
- Functionalities offered by Web Services similar to
  - traditional Remote Procedure Calls (RPC)
- To realize Web Services well known protocols are used, e.g.:
  - HTTP or SMTP to transport XML-based messages on L5
  - TCP on L4
  - IP on L3
- Web Services use XML for
  - data exchange and Remote Procedure Calls:
    - XML allows platform independent description of data
    - But performance drawback
      - because of conversion from/into XML

Web Service Technology

- Web Services is platform independent replacement for:
  - Microsoft DCOM
  - Java RMI
  - OMG CORBA / IIOP

- Web Services
  - are loosely coupled
  - can be dynamically discovered and invoked
  - are time independent,
    - supporting synchronous and asynchronous usage
  - location independent

6.1 Web Service Layer Model



6.2 SOAP – Simple Object Access Protocol

- SOAP
  - is a communication model specifying message exchange
    - i.e. is a XML-based specification of messages
  - used for communication with Web Services

- Content of the SOAP standard
  - Contains a syntax for the definition of XML-based messages
  - Defines rules for possible content of corresponding messages
  - Defines rules for message transport using various L5-protocols (HTTP, SMTP)
  - Conventions for Remote Procedure Calls

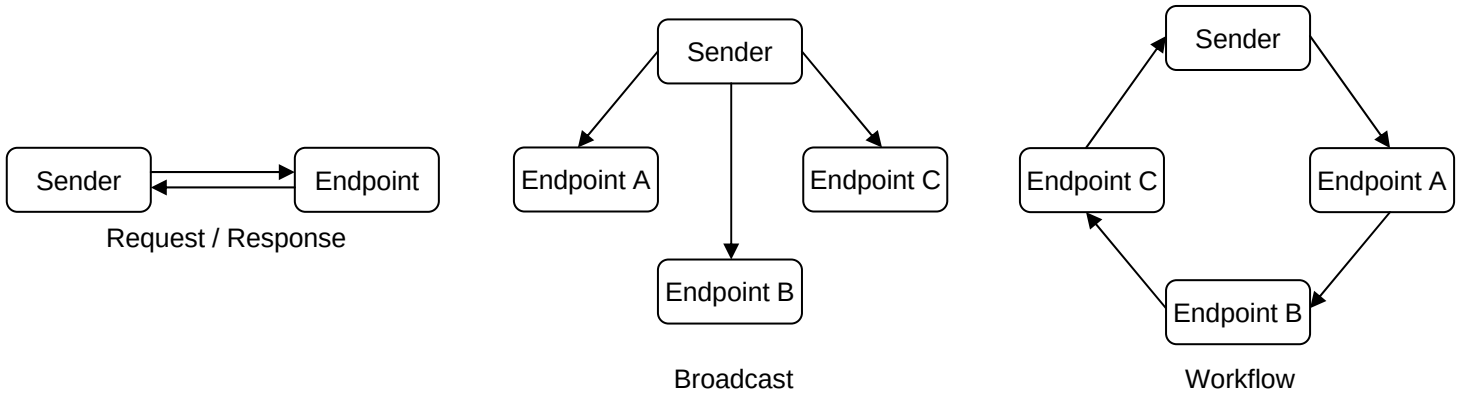
- Messages
  - are always exchanged between two participants
  - can be processed by different intermediaries on their way to the final receiver (so called endpoint)

- Every receiver (including intermediaries)
  - opens the messages and processes the part intended for him
- Every receiver
  - is able to be sender again so message passing is possible

SOAP – Simple Object Access Protocol

Possible communication models are:

- Request-response:
  - sender sends message, endpoint responds to message
- Broadcast:
  - sender generates messages concurrently transmitted to various receivers
- Workflow:
  - chain of senders and receivers processing a single task forming a circle



6.3 WSDL – Web Service Description Language

- WSDL is a specification
  - to describe interfaces of Web Services
  - using XML documents
- Usage of WSDL is obligatory
  - when using Web Services
- WSDL defines rules for method invocation
  - (so called contracts)

- WSDL document describing a Web Service comprises
  - Definition of data types
  - Declaration of methods
  - Combination of methods to Web Services
  - Mapping of protocols for method invocation (binding)

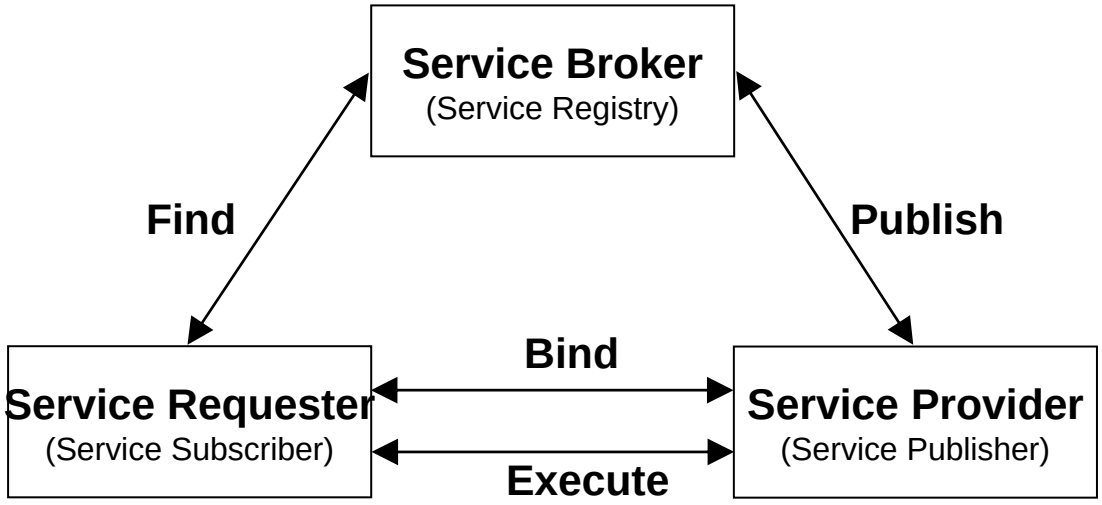
- WSDL allows the following bindings to be used:
  - SOAP
  - HTTP GET/POST
  - MIME

- Possible types of communication defined by WSDL:
  - One-Way:
    - clients sends message to server, no response by server
  - Request-Response:
    - client sends message to server, server sends response
  - Solicit-Response:
    - server sends message to client, response by client
  - Notification:
    - server sends message to client, no response by client

6.4 UDDI – Universal Description, Discovery and Integration

- UDDI contains various aspects needed to support dynamic access and usage of Web Services:
  - A logically unique but physically distributed directory service architecture
    - in which Web Services can be published and searched
  - Requirements on providers of these directory services
  - A description of an API enabling
    - the publishing and
    - searching of Web Services
  - An XML-based data model
    - to describe companies offering Web Services and
    - to describe the Web Services itself
- Contents of the UDDI data model are (not limited to Web Services):
  - White Pages:
    - information about the company offering the Web Service
  - Yellow Pages:
    - classification of the company offering the Web Service
  - Green Pages:
    - description of the offered Web Service,
    - containing technical information
      - how to find and use the offered Web Service

6.5 Publish-Find-Bind-Execute Paradigm



Roles in a Web Service based architecture:

- Service Requestor
- Service Provider
- Service Registry (optional)

Interaction between roles (including Service Registry):

- Service Provider registers Web Service with Service Broker (using UDDI).
- Service Requestor
  - searches for Web Service in UDDI registry and
  - receives information about where to find and how to use Web Service (via UDDI API).
- Service Requestor
  - binds Service based on the information given by the Service Broker (based on WSDL).
  - is able to execute Services provided by Service Provider (using SOAP).