# Management of End-Systems

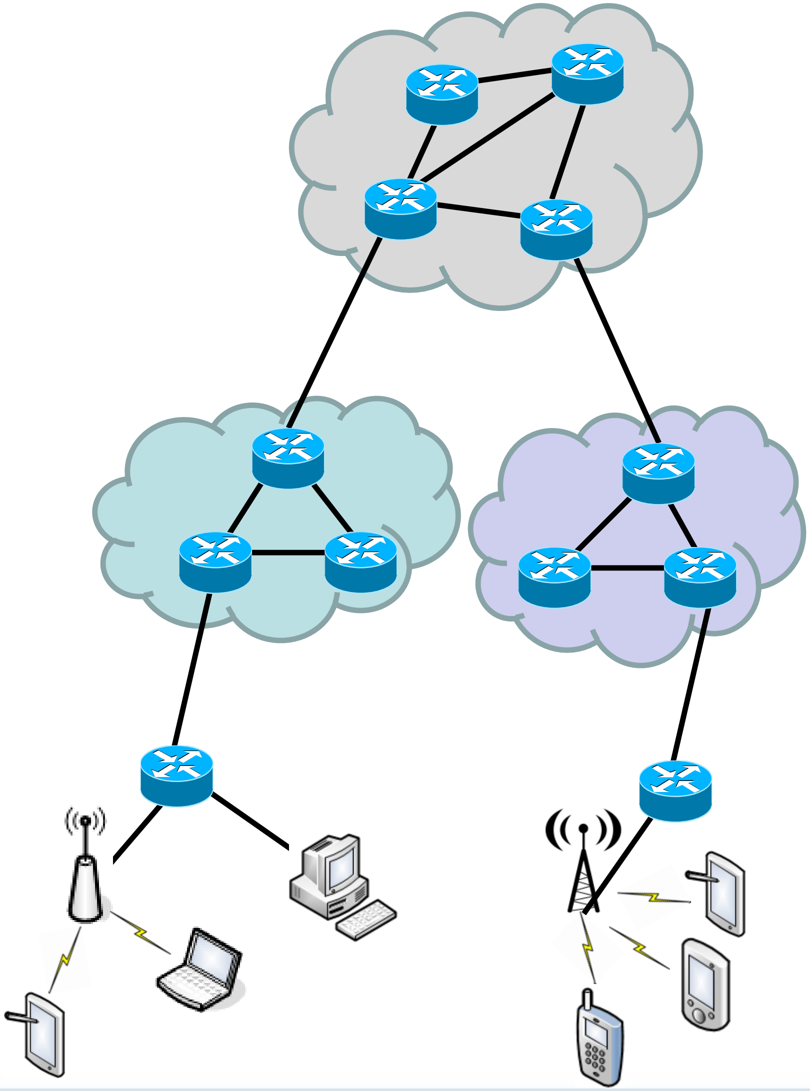
Network Management

Prof. Dr. Panagiotis Papadimitriou

# End Systems



- End-systems:
  - Desktops, laptops, PDAs, other mobile terminals
  - Run applications (web, email, voice, video, etc.)
- Two service models:
  - Client / server
  - Peer-to-peer



# End-System Management



- Network discovery and bootstrapping
  - Joining the network
  - Obtaining a network address
- Interface to network applications
  - Binding network applications to the transport layer
  - Identifying applications from port numbers
  - Port scanning
- Remote Procedure Calls (RPC)
  - XML-RPC

# Network Discovery and Bootstrapping

# Naming and Addressing



- Network identifiers:
  - Host Name (e.g., www.uni-hannover.de)
  - IP address (e.g., 172.23.180.137)
  - MAC address (e.g., 84-2B-2B-A5-A5-77)

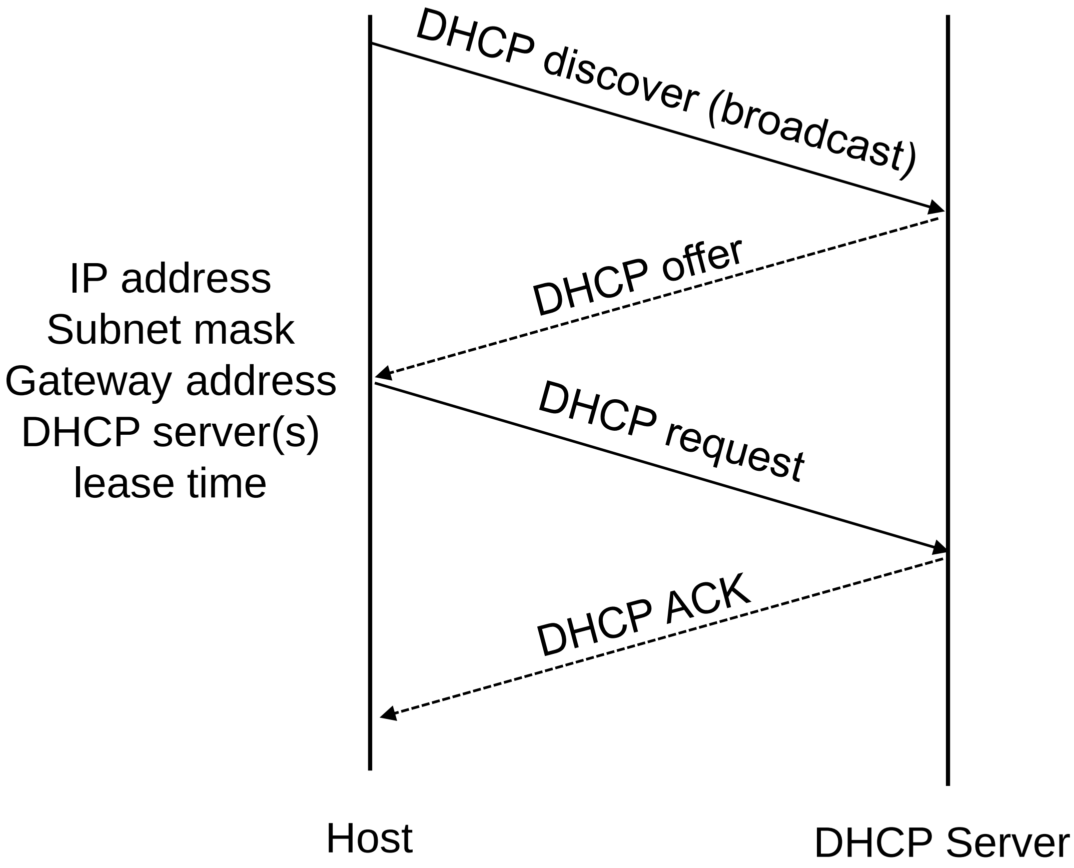
- Mapping between identifiers:
  - Address Resolution Protocol (ARP)
    - Physical address resolution
  - Domain Name System (DNS)
    - Host name resolution

- Dynamic IP address assignment
  - Dynamic Host Configuration Protocol (DHCP)

# Dynamic Host Configuration Protocol (DHCP)



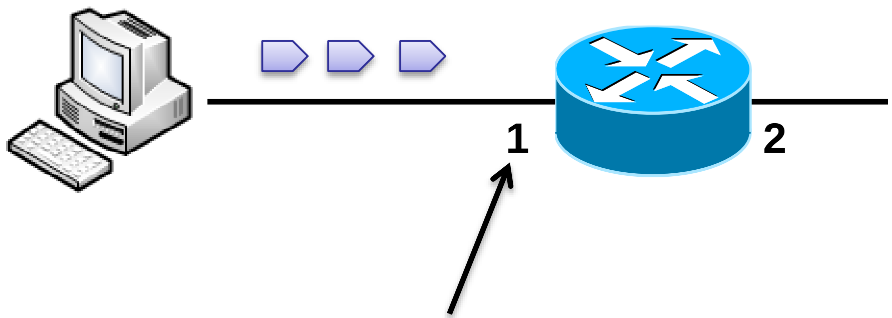
- DHCP is used for dynamic network configuration of end systems



# Physical Address Resolution



- Sending IP packets requires:
  - the IP address of the next-hop (provided by the forwarding table)
  - the physical address of the next-hop




Next-hop interface (IP, MAC???)

| IP Header | Payload |
|---|---|

IP packet encapsulation

| Frame Header | IP Header | Payload |
|---|---|---|

# Address Resolution Protocol (ARP)

- ARP is responsible for resolving physical addresses:
  - ARP allows routers/hosts in the network to build up a table of mappings between IP and physical addresses
  - each entry in the ARP table is removed after a time period (TTL)

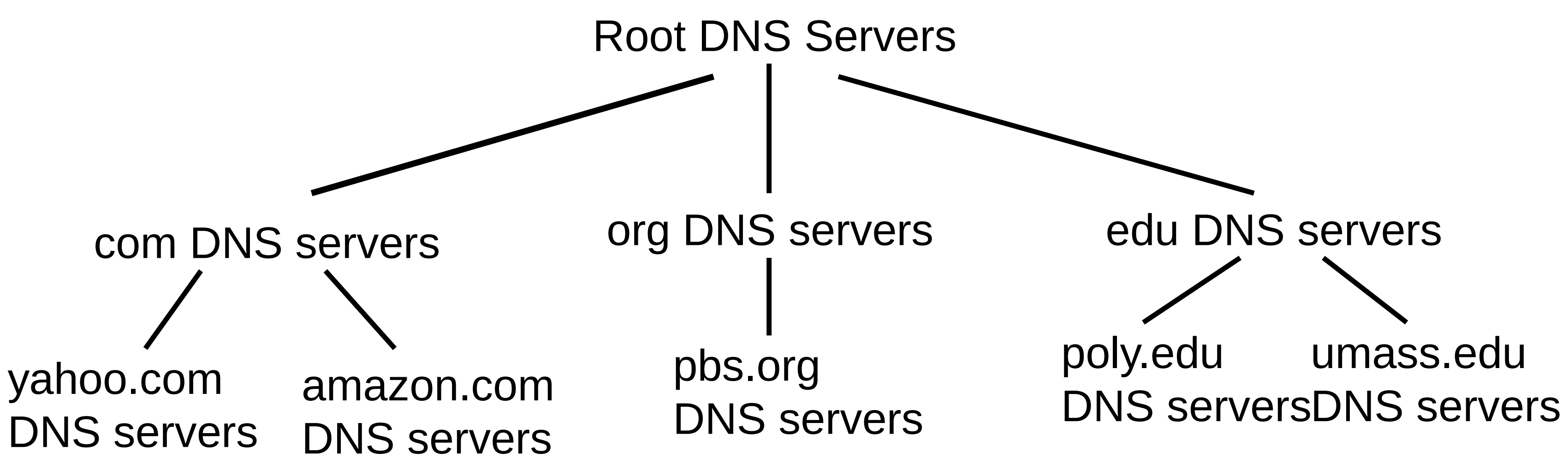
| IP Address | Physical Address | TTL |
|---|---|---|
| 172.23.180.137 | 84-2B-2B-A5-A5-77 | 13:45:00 |
| 172.23.42.42 | 70-F3-95-3D-14-04 | 11:12:00 |

- When a host/router wants to identify a physical address:
  - it performs a lookup in the ARP table and obtains the physical address
  - if there is no match:
    - it broadcasts an ARP query onto the network
    - the host/router that has this IP address sends a message with its physical address (which is subsequently added to the ARP table)

# Domain Name System (DNS)



- DNS provides hostname to IP address translation
- DNS uses an hierarchical, distributed database



# DNS Servers

- Local DNS servers:
  - the first contact point for hostname resolution
  - forward queries to root DNS servers
  - cache hostname mappings locally
    - cache entries expire after a certain period

- Root DNS servers:
  - contacted by local DNS servers that cannot resolve hostname

- Top-level Domain (TLD) servers:
  - responsible for com, org, edu, etc. and all top-level country domains (e.g., de, uk, fr, jp)

- Authoritative DNS servers:
  - belong to organizations (e.g., enterprises, universities)
  - provide hostname resolution for organization’s servers (e.g., web, e-mail)

# Root DNS Server Locations



13 worldwide root servers

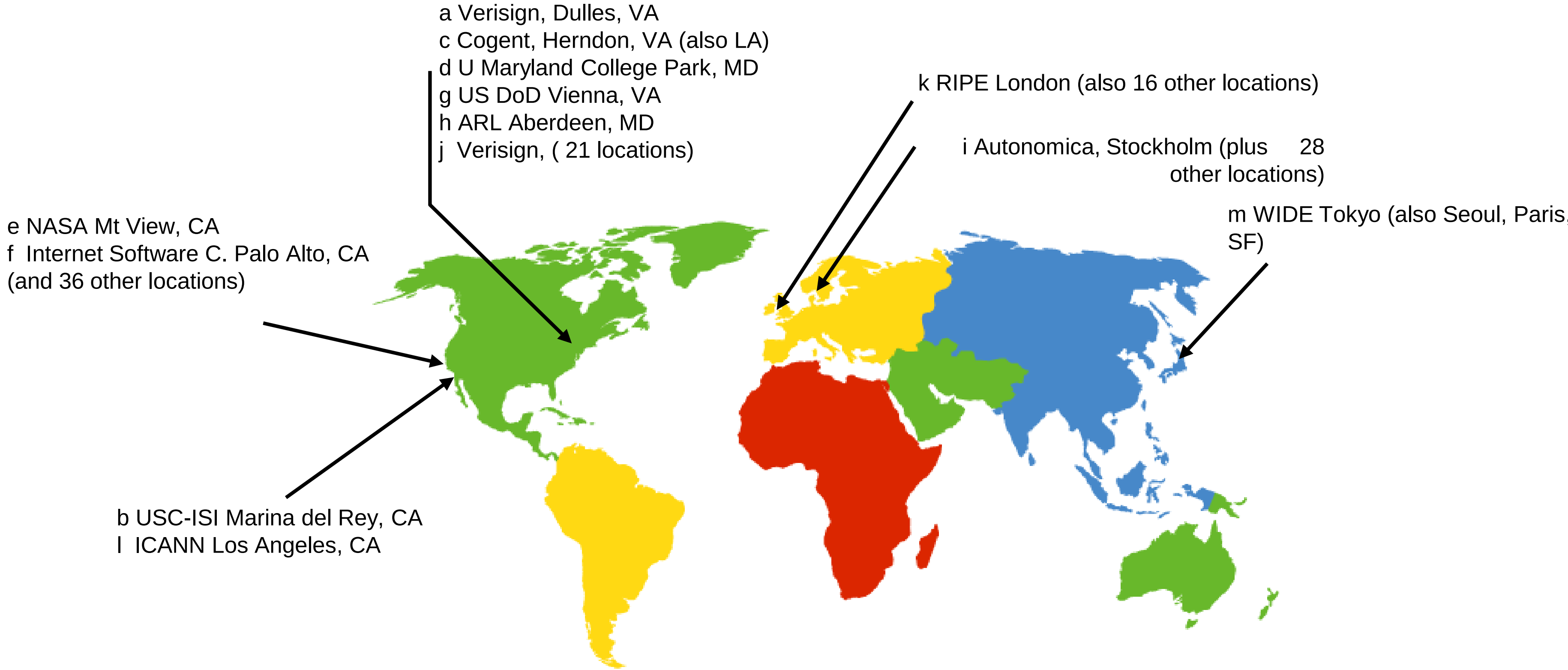
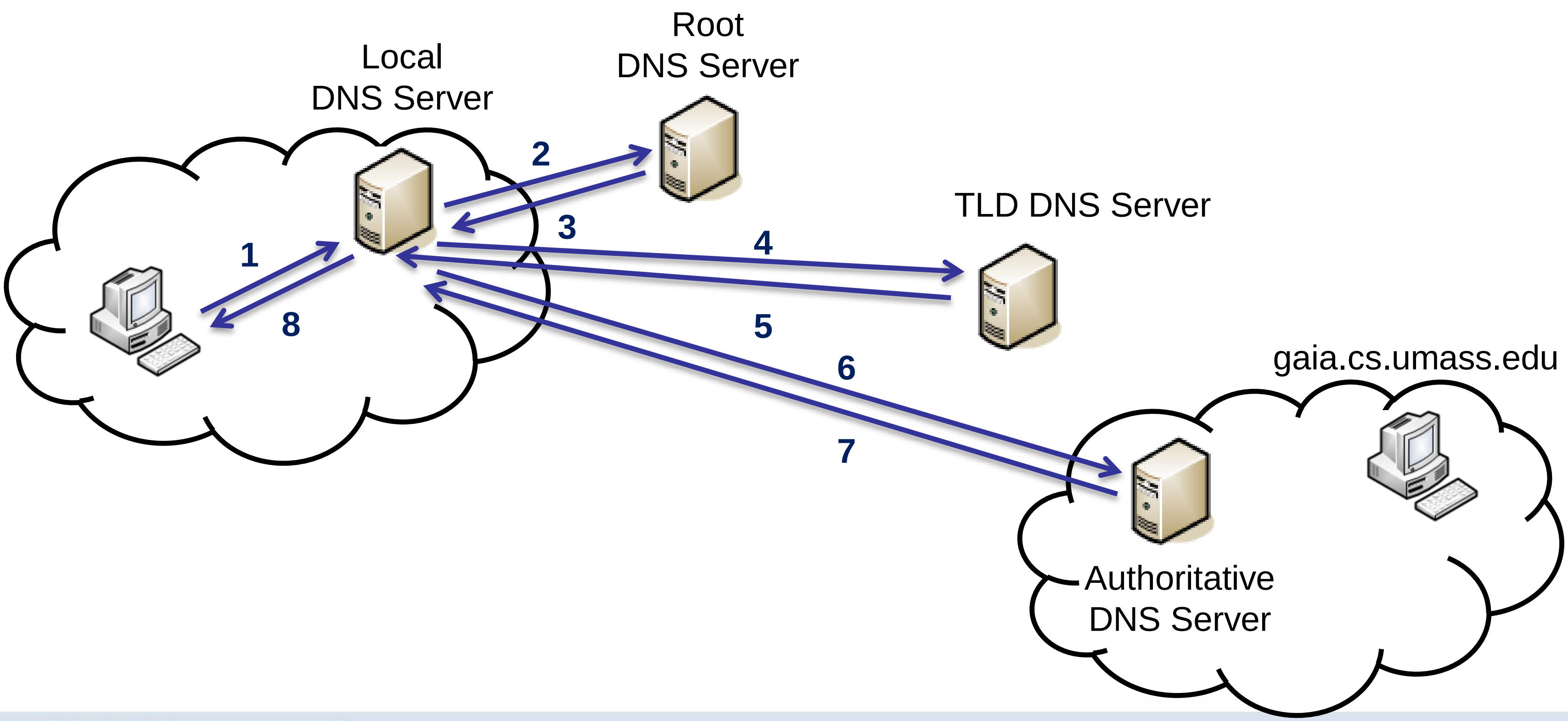



Figure from Computer Networking, A Top-Down Approach

# DNS Name Resolution



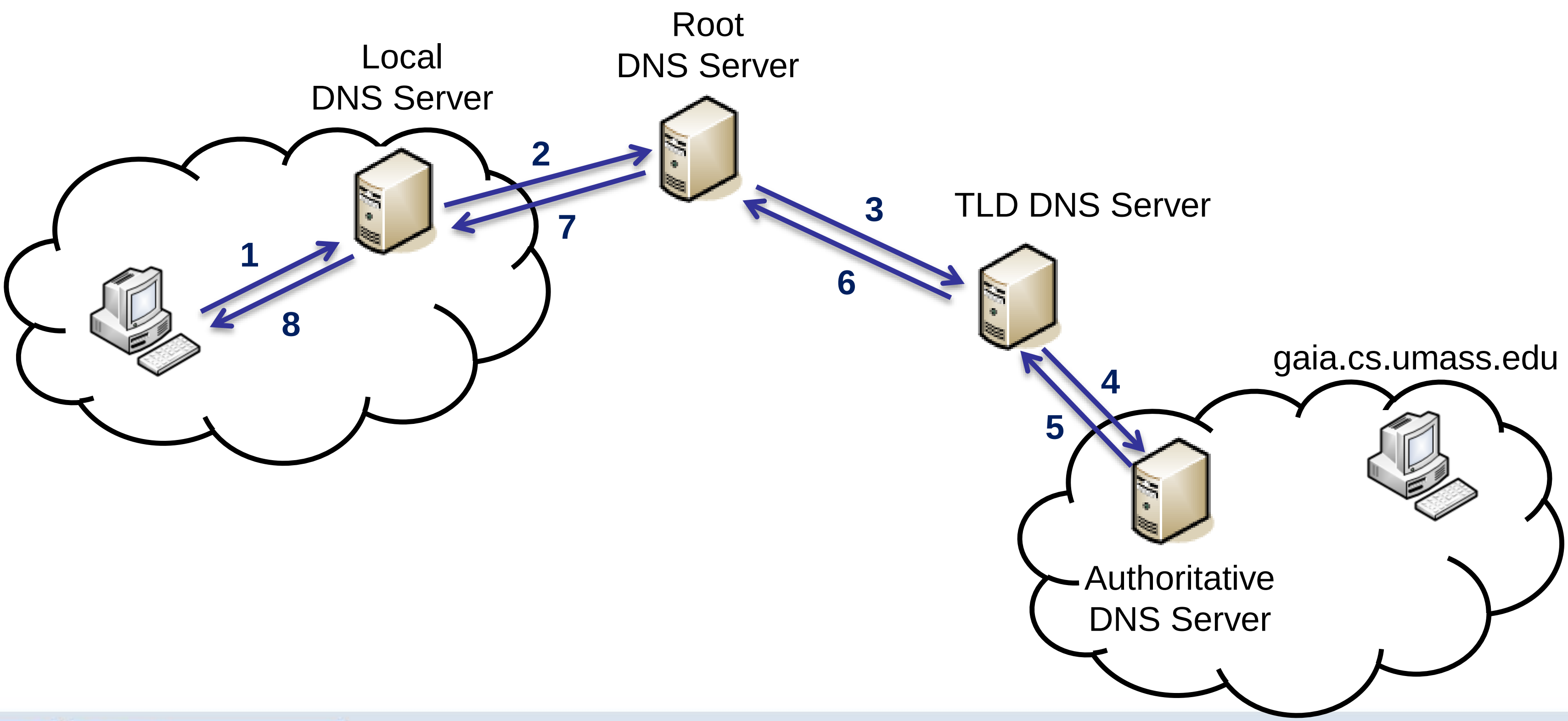
- The host **pc1.ikt.uni-hannover.de** queries the IP address for **gaia.cs.umass.edu**:
  - Iterated query



# DNS Name Resolution

Institut für Kommunikations-Technik

- The host **pc1.ikt.uni-hannover.de** queries the IP address for **gaia.cs.umass.edu**:
  - Recursive query



# DNS Records



- DNS record format: `[name, value, type, TTL]`
- DNS record types:
  - Address (A) record for hosts
    - e.g., `[relay1.bar.example.com, 123.45.120.12, A, TTL]`
  - Mail Exchanger (MX) record for email servers
    - e.g., `[example.com, mail.example.com, MX, TTL]`
  - Name Server (NS) record for authoritative DNS servers of a domain
    - e.g., `[example.com, dns.example.com, NS, TTL]`
  - Canonical Name (CNAME) record:
    - `name` is an alias for a “canonical” name
    - e.g., `[example.com, relay1.bar.example.com, CNAME, TTL]`
    - allows the use of mnemonic hostnames
    - widely used for the redirection of client requests to a proximate server in content distribution networks (CDNs)

# DNS Resolver Configuration



```
$TTL 86400
my-name.com.            IN      SOA     debns1.my-name.com. \
                                        joe.my-name.com. {
                    2004011522      ; Serial no., based on date
                         21600      ; Refresh after 6 hours
                          3600      ; Retry after 1 hour
                        604800      ; Expire after 7 days
                          3600      ; Minimum TTL of 1 hour
)
;Name servers
debns1                  IN      A        192.168.1.41
debns2.joescuz.com.     IN      A        192.168.1.42

@                       IN      NS       debns1
my-name.com.            IN      NS       debns2.my-name.com.


;Mail servers
debmail1                IN      A        192.168.1.51
debmail2.my-name.com.   IN      A        192.168.1.52

@                       IN      MX       10 debmail1
my-name.com.            IN      MX       20 debmail2.my-name.com.

;Aliased servers
debhp                   IN      A        192.168.1.61
debdell.my-name.com.    IN      A        192.168.1.62

www                     IN      CNAME    debhp
ftp.my-name.com.        IN      CNAME    debdell.my-name.com.
```

# DNS Extensions



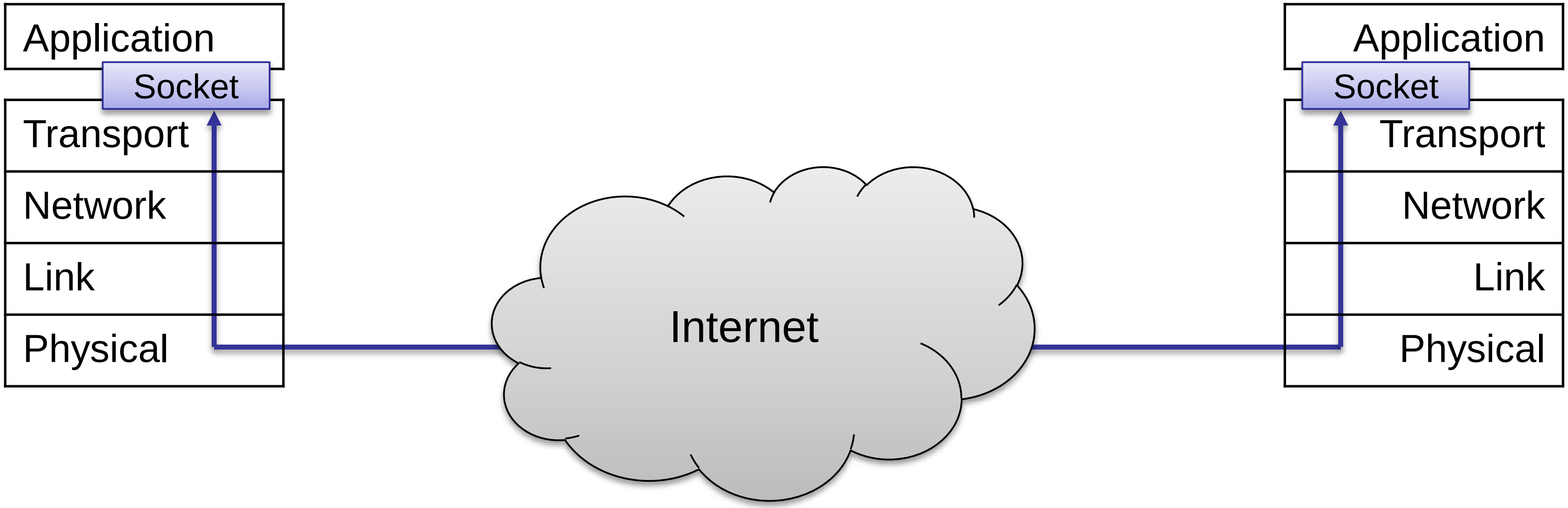
- DNSSEC (Domain Name System Security Extensions):
  - provides authentication of DNS data
    - DNS responses contain a DNSSEC certificate
  - deployed on root DNS servers

- edns-client-subnet:
  - includes part of the client IP address in the DNS request
    - allows the selection of a CDN server in proximity to the client (instead of the client's name resolver)

# Interface to Network Applications

# Sockets

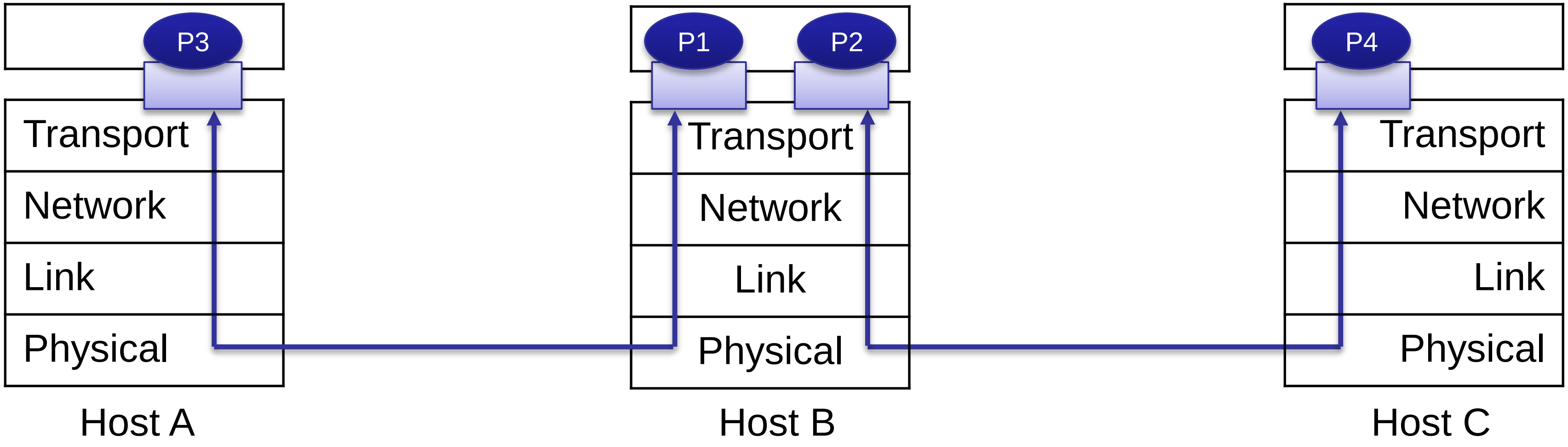
- Sockets provide an interface between applications and transport protocols



# Demultiplexing with Sockets



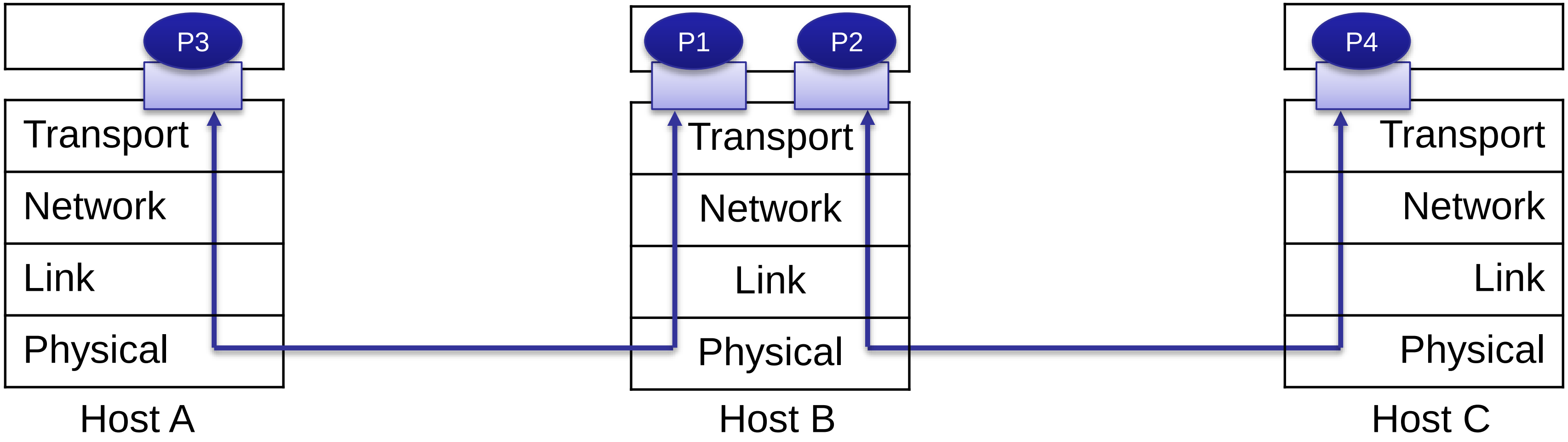
- Host uses IP address and port number to deliver a segment to the appropriate socket



# Demultiplexing with TCP Sockets



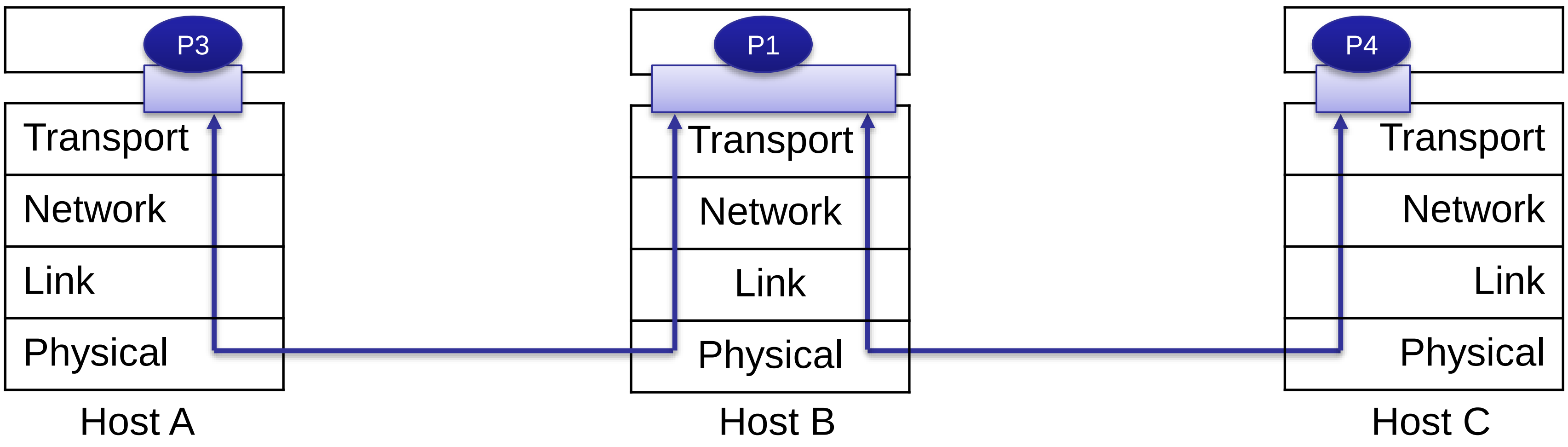
- TCP sockets are identified by a 4-tuple:
  - Source IP address
  - Source port number
  - Destination IP address
  - Destination port number
- A host may support multiple TCP sockets (e.g., host B):
  - Each socket is identified by its own 4-tuple



# Demultiplexing with UDP Sockets

Institut für Kommunikations-Technik

- UDP sockets are identified by a 2-tuple:
  - Destination IP address
  - Destination port number
- A host supports one UDP socket per port



# UDP Server and Client Socket Using Python



```
# UDP server socket
import socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind(("", 5000))

print "UDP server waiting for client on port 5000"

while 1:
   data, address = server_socket.recvfrom(256)
   print "( " ,address[0], " " , address[1] , " ) said : ", data
```

```
# UDP client socket
import socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
client_socket.sendto("Hello", ("125.10.37.1",5000))
client_socket.close()
```

# Port Numbers

Institut für Kommunikations-Technik

- Port numbers represented by 16-bit unsigned integers (0 – 65535)
- Classification of port numbers:
  - Well-known ports:
    - Range: 0 – 1023
    - Only for root-privileged applications (e.g., SSH)
  - Registered ports:
    - Range: 1024 – 49151
  - Dynamic and private ports:
    - Range: 49152 – 65535

# Applications and Port Numbers

| Application | Protocol | Port Number |
|---|---|---|
| FTP | TCP | 20, 21 |
| SSH | TCP / UDP | 22 |
| Telnet | TCP | 23 |
| SMTP | TCP | 25 |
| DNS | UDP / UDP | 53 |
| HTTP | TCP / UDP | 80 |
| POP3 | TCP | 110 |
| NTP | UDP | 123 |
| IMAP | TCP | 143 |
| SNMP | UDP | 161 |
| SNMP Trap | TCP | 162 |
| DHCP | TCP / UDP | 546, 547 |

# Identifying Applications (1)

- Identifying applications from port numbers may be inaccurate:
  - Some (new) applications may not use well-known port numbers
  - Applications may tunnel their traffic over HTTP to go through firewalls (e.g., tunneling SSH over HTTP)

- Accurate application classification requires further inspection:
  - Packet header inspection
  - Deep-packet inspection for well-known signatures or protocol semantics

- Duration of inspection may vary depending on the classification technique:
  - 1st packet
  - 1st Kbyte
  - Flow

# Identifying Applications (2)

| Classification | Example Application |
|---|---|
| BULK | ftp |
| DATABASE | postgres, sqlnet, oracle, ingres |
| INTERACTIVE | ssh, klogin, rlogin, telnet |
| MAIL | imap, pop2/3, smtp |
| SERVICES | X11, dns, ident, ldap, ntp |
| WWW | www |
| P2P | KaZaA, BitTorrent, GnuTella |
| MALICIOUS | Internet work and virus attacks |
| GAMES | Half-Life |
| MULTIMEDIA | Windows Media Player, Real |

| Classification Type | Port-Based | | Content-Based | |
|---|---|---|---|---|
| | Packets | Bytes | Packets | Bytes |
| | As a percentage of total traffic | | | |
| BULK | 46.97 | 45.00 | 65.06 | 64.54 |
| DATABASE | 0.03 | 0.03 | 0.84 | 0.76 |
| GRID | 0.03 | 0.07 | 0.00 | 0.00 |
| INTERACTIVE | 1.19 | 0.43 | 0.75 | 0.39 |
| MAIL | 3.37 | 3.62 | 3.37 | 3.62 |
| SERVICES | 0.07 | 0.02 | 0.29 | 0.28 |
| WWW | 19.98 | 20.40 | 26.49 | 27.30 |
| UNKNOWN | 28.36 | 30.43 | <0.01 | <0.01 |
| MALICIOUS | — | — | 1.10 | 1.17 |
| IRC/CHAT | — | — | 0.44 | 0.05 |
| P2P | — | — | 1.27 | 1.50 |
| GAMES | — | — | 0.17 | 0.18 |
| MULTIMEDIA | — | — | 0.22 | 0.21 |

A. Moore and D. Papagiannaki, “Toward the Accurate Identication of Network”, PAM 2005

# Port Scanning



- Port scanning allows attackers to discover open ports on a certain host:
  - Ports characterize the amount of exposure to external attacks
  - A system administrator should be aware of processes listening on ports and possible weaknesses arising by that

- Various port scanning techniques:
  - Full TCP connection scanning
  - TCP SYN scanning
  - TCP FIN scanning
  - Indirect scanning
  - UDP scanning

# TCP Header

Institut für Kommunikations-Technik

32 bits

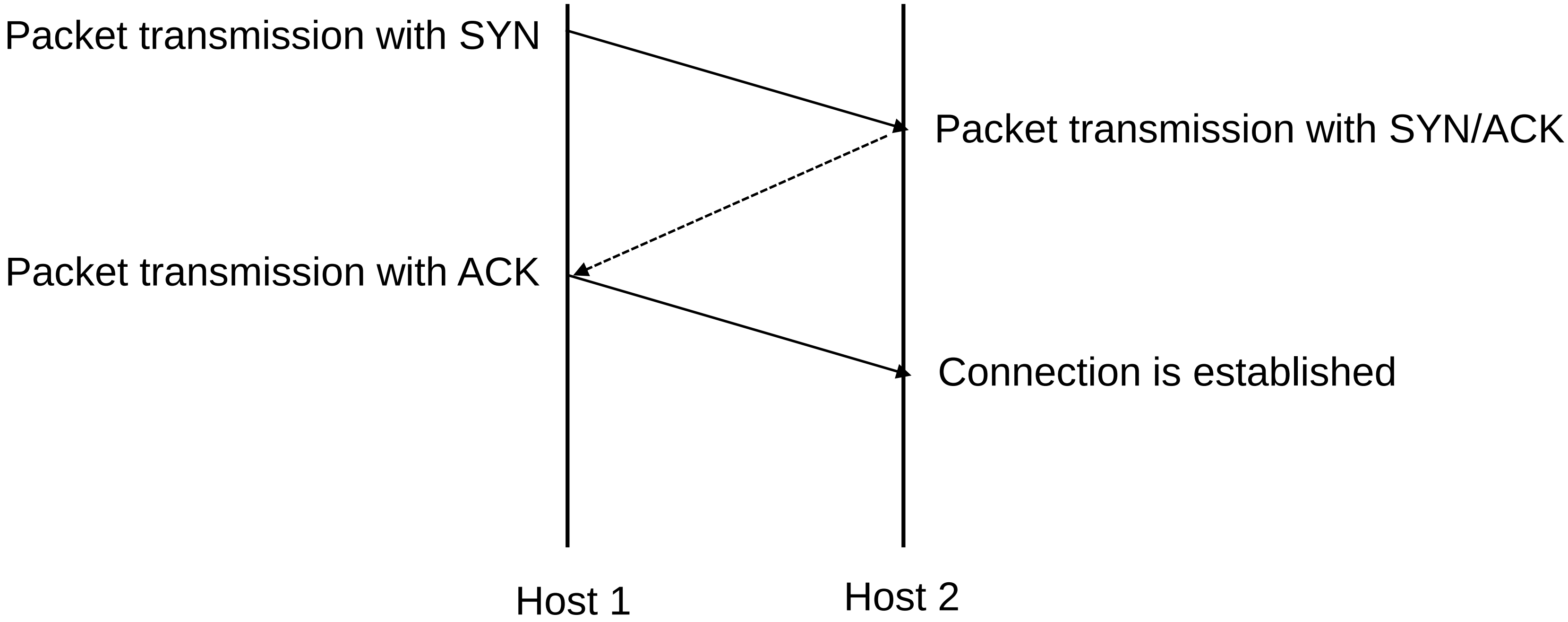
20 bytes

<table>
<tr><td colspan="8">Source Port</td><td>Destination Port</td></tr>
<tr><td colspan="9">Sequence Number</td></tr>
<tr><td colspan="9">Acknowledgment Number</td></tr>
<tr><td>Header Length</td><td>Unused</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>Receiver window</td></tr>
<tr><td colspan="8">Checksum</td><td>Urgent data pointer</td></tr>
<tr><td colspan="9">Options (variable length)</td></tr>
<tr><td colspan="9">Data</td></tr>
</table>

# TCP Connection Establishment

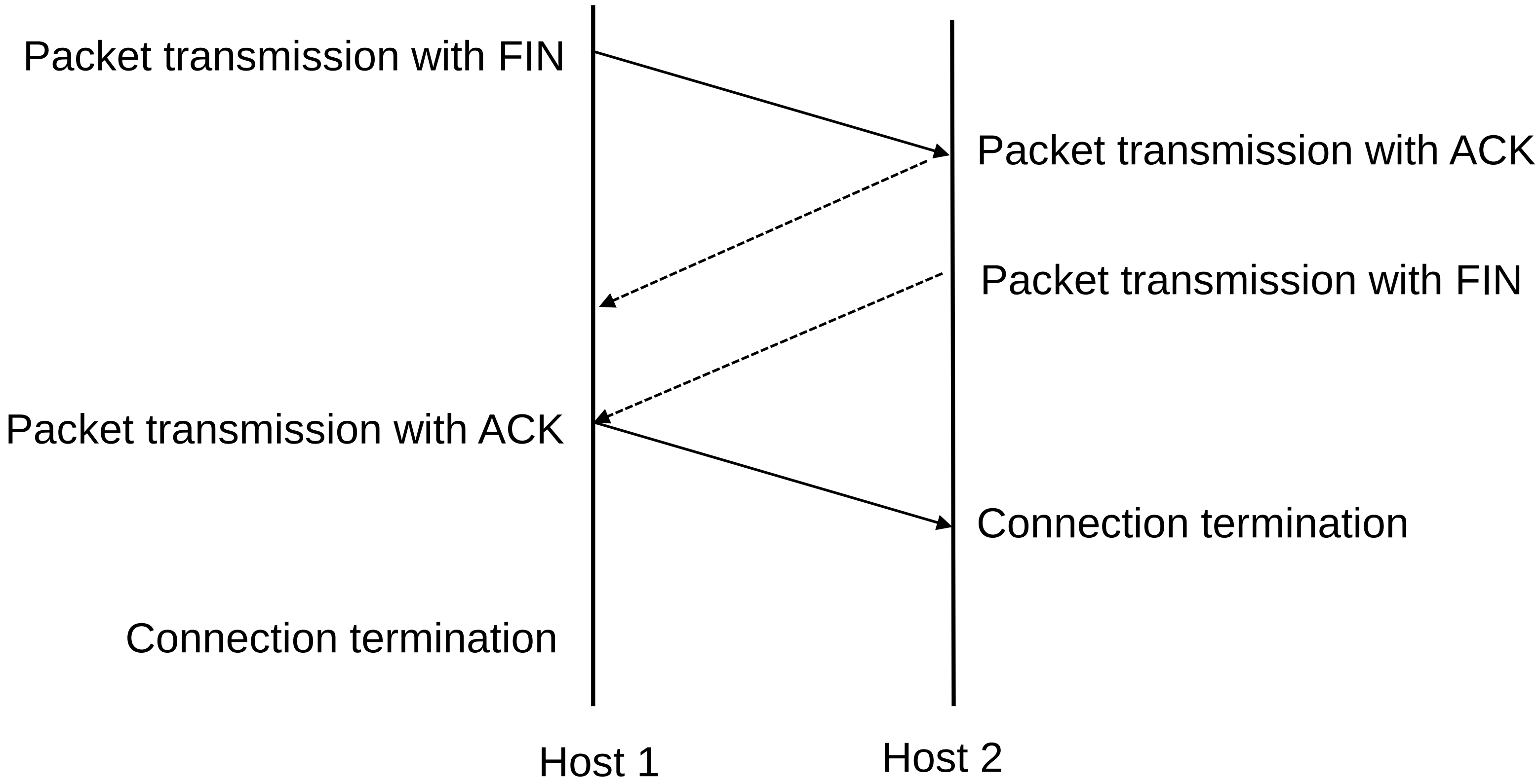


- TCP uses a “3-way handshake” to establish connection



# TCP Connection Termination





# Full TCP Connection Scanning

- Try to setup connections with chosen ports on the target host:
  - Connection is established for every listening port
  - Otherwise, the port is unreachable

- Full TCP connection scanning:
  - Does not need any root privileges and can be performed by any user
  - Can be easily detected
  - Slow

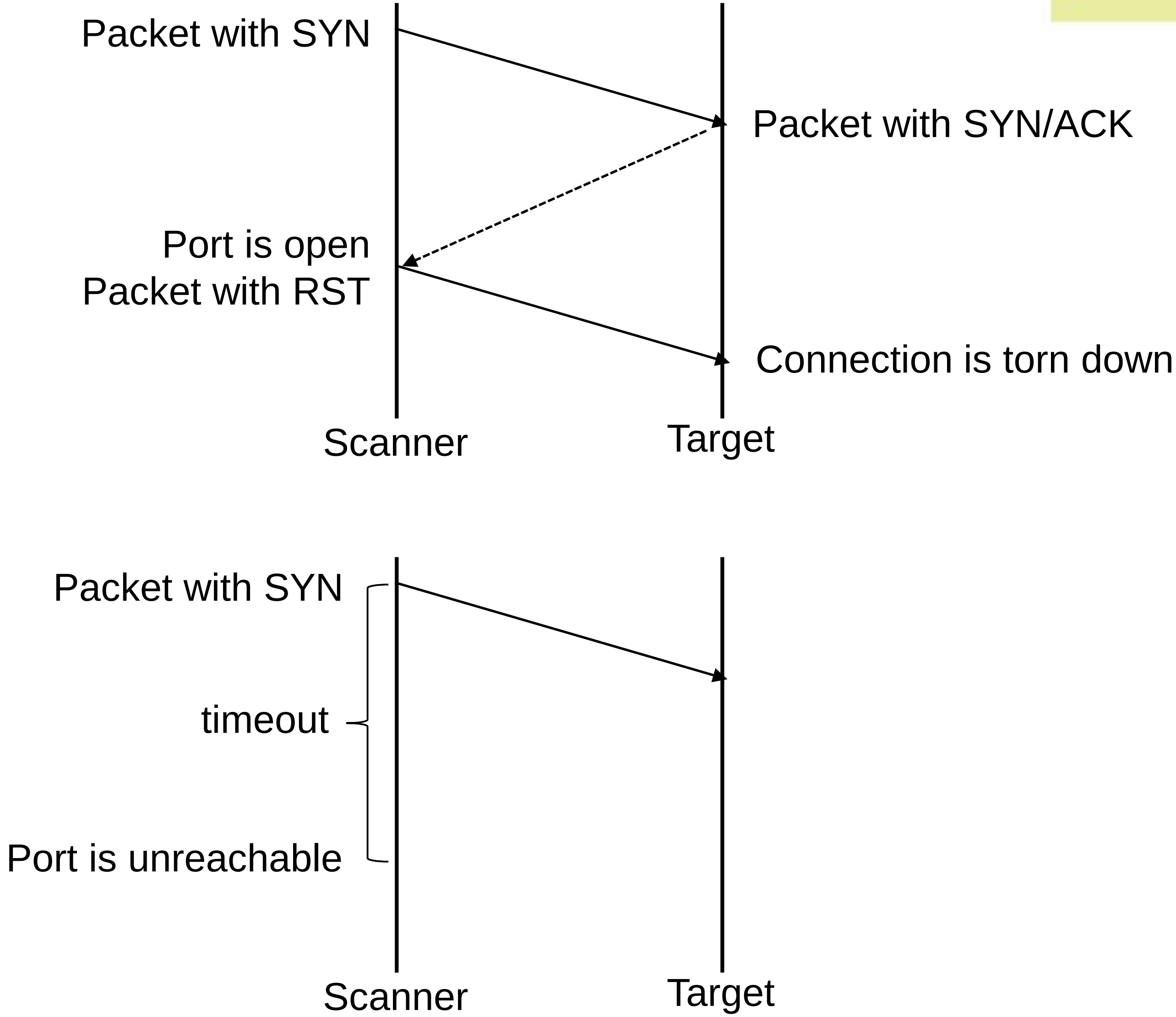
# TCP SYN Scanning

- Try to setup half-open TCP connections with the target host:
  - The scanning host sends a SYN packet and waits for the response
  - Reception of SYN/ACK indicates that the port is listening
    - The scanning host sends an RST to tear down the connection immediately
  - No response indicates that the port is unreachable

- TCP SYN scanning:
  - Requires root privileges
  - Hosts may not detect this type of port scanning

# TCP SYN Scanning





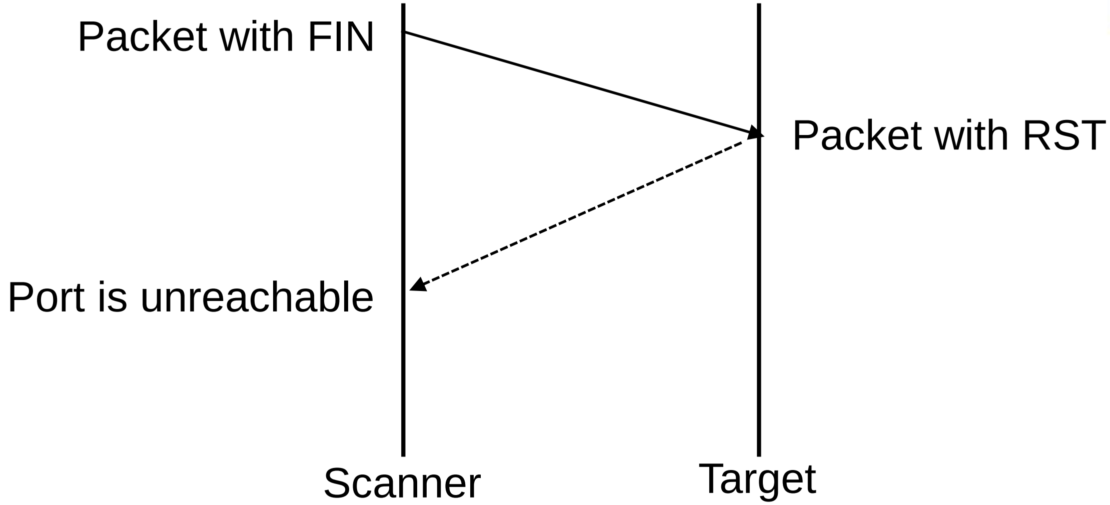
# TCP FIN Scanning

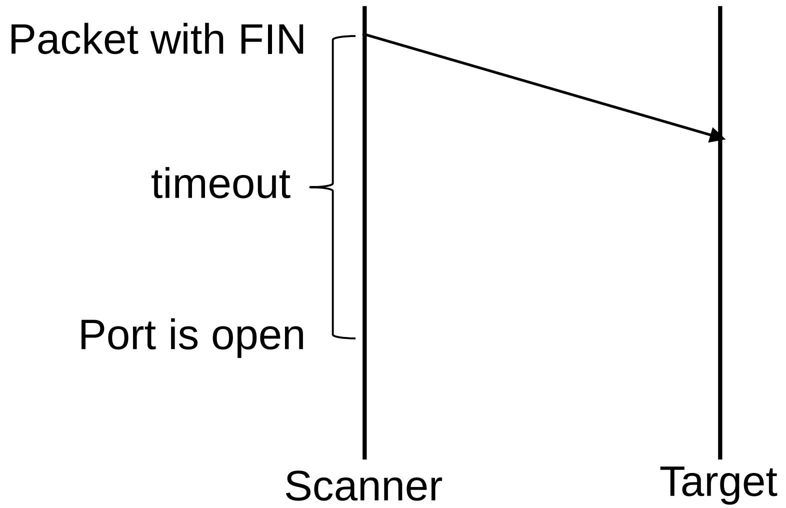
Institut für Kommunikations-Technik

- Use FIN packets to probe ports on the target host:
  - The scanning host sends a FIN packet and waits for the response
  - If the port is closed, the target host will drop the FIN packet and respond with an RST packet
  - In the case of a listening port, the target host will drop the FIN packet without sending a response
- TCP FIN scanning:
  - Requires root privileges
  - Hosts may not detect this type of port scanning
  - Some hosts may not be vulnerable to TCP FIN scans, since they return RST packets regardless of the port state

# TCP FIN Scanning





# UDP Scanning



- Detection of open UDP ports on the target host:
  - The scanning host sends a UDP packet and waits for the response
  - Reception of an ICMP error message indicates that the port is unreachable
  - No response indicates that the port is open
- UDP port scanning:
  - Useful for the detection of phony servers or unauthorized servers (e.g., DNS, NFS, SMTP)

# Detecting Port Scanning

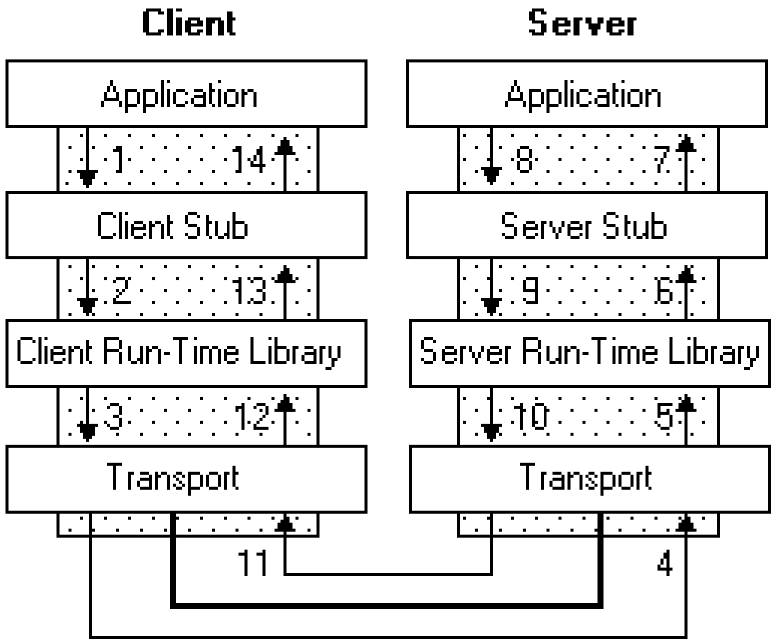


- System administrator may use several applications and loggers to detect port scans:
  - Courtney
  - Gabriel
  - scan_detector
  - TCP Wrapper
  - scanlogd
  - Argus
  - tcplogger

# Remote Procedure Calls

# Remote Procedure Calls (RPC)

- Remote procedure calls (RPC) make a call to a remote system appear as a local call:
  - Whether the server is local or remote is made transparent by RPC
  - RPC allows applications to become distributed transparently
  - RPC makes the architecture of the remote system transparent



# Stubs

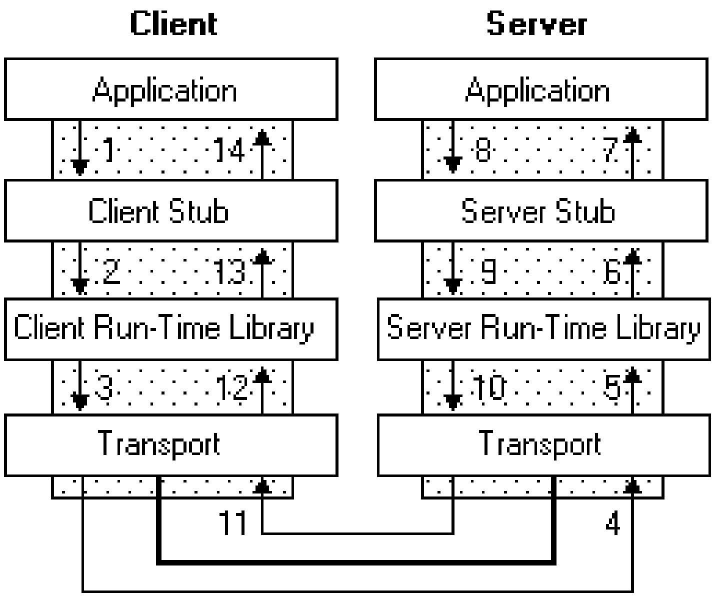


- Stubs run in the client and server carrying out data conversion, as needed:
  - The client and server use different address spaces
    - Pointers to one computer’s memory will point to different data in a remote machine
  - The client and server may use different data representations even for simple parameters
    - e.g., big-endian vs. little-endian

# RPC Client Functionality

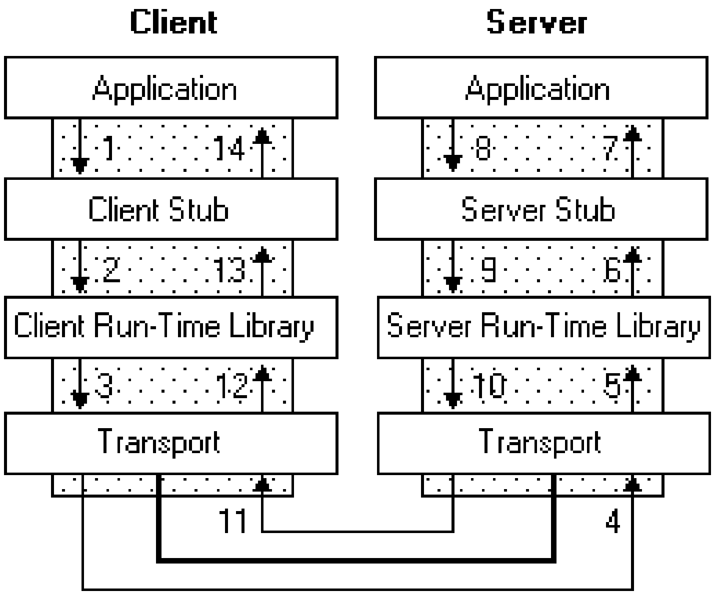
Institut für Kommunikations-Technik

- The client performs the following steps to initiate the RPC at the server:
  1. Retrieves the required parameters from the client address space
  2. Translates the parameters into the required format for transmission over the network
  3. Calls functions in the RPC client run-time library to send the request and its parameters to the server



# RPC Server Functionality (1)

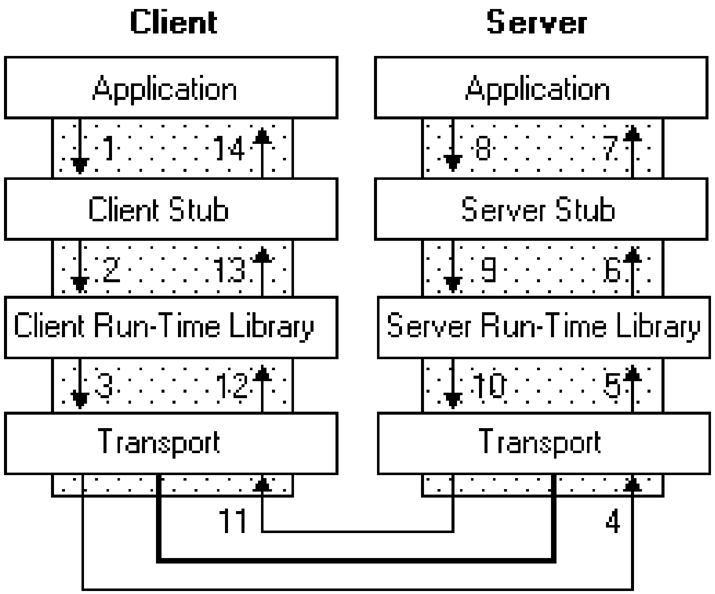
- The server performs the following steps to call the procedure:
  5. The server RPC run-time library functions accept the request and call the server stub procedure
  6. The server stub retrieves the parameters from the network buffer and converts them from the network transmission format to the format the server needs
  7. The server stub calls the actual procedure on the server



# RPC Server Functionality (2)

- When the remote procedure is complete, a similar sequence of steps returns the data to the client:

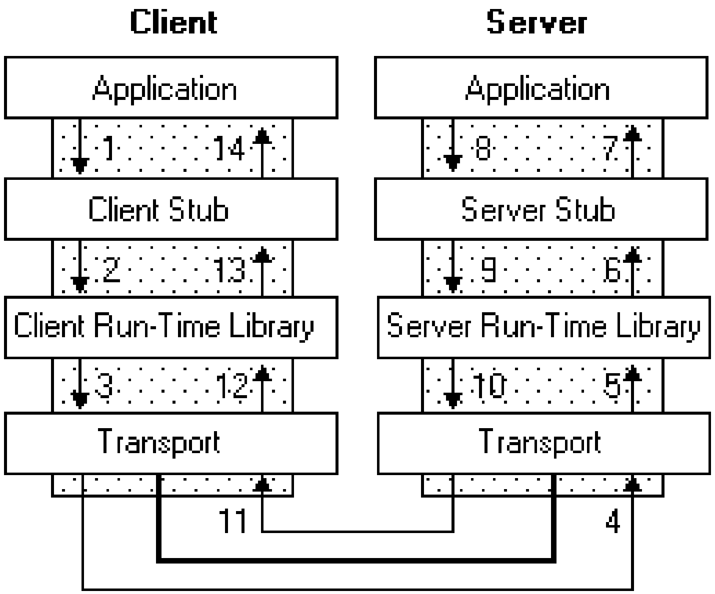
8. The remote procedure returns its data to the server stub
9. The server stub converts output parameters to the format required for transmission over the network and returns them to the RPC run-time library functions
10. The server RPC run-time library functions transmit the data on the network to the client computer



# RPC Client Functionality



- The client completes the process by accepting the data over the network and returning it to the calling function:
  12. The client RPC run-time library receives the remote-procedure return values and returns them to the client stub
  13. The client stub converts the data into the format used by the client computer. The stub writes data into the client memory and returns the result to the calling program on the client
  14. The calling procedure continues as if the procedure had been called on the same computer



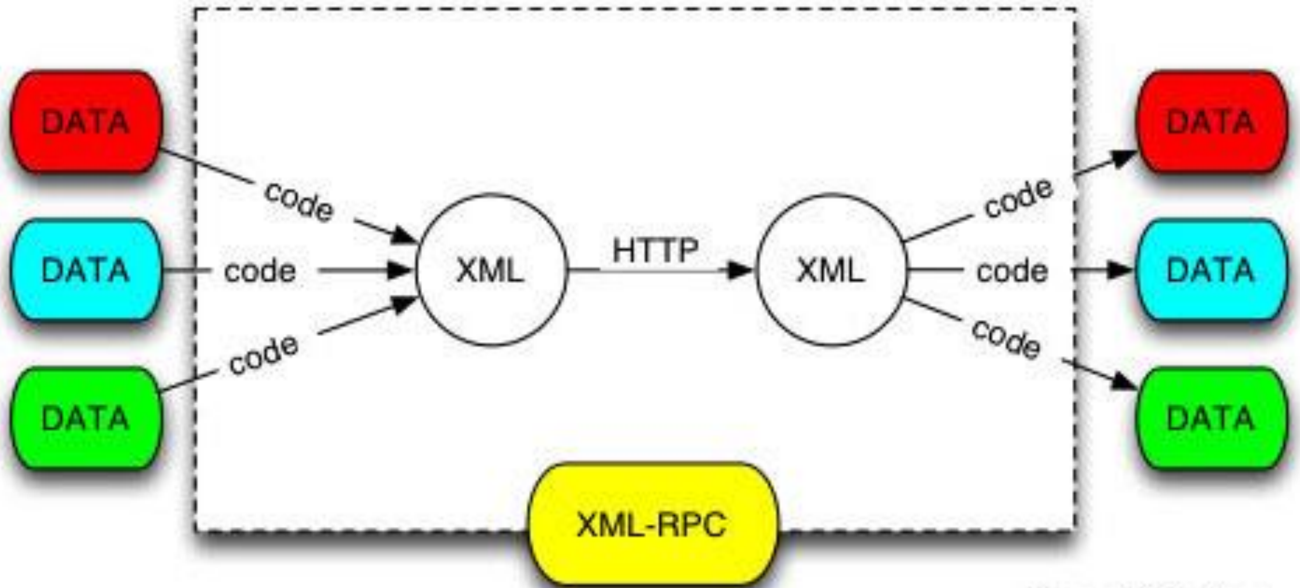
# RPC in Management of End-systems



- RPC can facilitate the management of end-systems:
  - Collection of management data from end-systems
    - Store and retrieve management data from a local database
  - Network monitoring
    - Periodically gather operational data
    - Compare previously collected data against the most recent
  - Device configurations
    - Add / modify / delete device configurations (e.g., network interfaces)
  - Fault management and diagnosis

# XML-RPC

- XML-RPC is an RPC protocol which uses:
  - HTTP for data transport
  - XML for data encoding




Source: JY Stervinou

# XML-RPC Data Types



| Data Type | Description |
|---|---|
| integer (i4) | 32-bit signed integer |
| boolean | boolean (0 or 1) |
| string | string |
| double | 64-bit signed, floating point number |
| dateTime.iso8601 | pseudo ISO8601 timestamp,(e.g., 19980717T14:08:55). Compared to ISO8601, milliseconds and time zone information is missing |
| base64 | base64 encoded binary data |
| struct | key-value pair. Keys are strings and values can be any valid data type |
| array | array of objects. The array elements can be any valid data type |

# Extensible Markup Language (XML)



- XML provides a self-describing data format:
  - Application reads data, parses it, and knows exactly what each constituent part of the data means

- Strengths of XML:
  - Easy to understand, parse and debug
  - Can handle complex data
  - Wide range of available tools
  - Extensibility and interoperability

# XML-based Description of Network Topology

Institut für Kommunikations-Technik

```
<network id="net1">
  <link id="1">
    <description type="link">
    <host1 id="node1" />
    <host2 id="node5" />
    <interface1 id="interface1" />
    <interface2 id="interface0" />
    <bandwidth id="1024" />
    </description>
  </link>
          :
          :
          :
          :
  <link id="10">
    <description type="link">
    <host1 id="node3" />
    <host2 id="node8" />
    <interface1 id="interface1" />
    <interface2 id="interface1" />
    <bandwidth id=“850" />
    </description>
  </link>

</network>
```

# References


- A. Moore and D. Papagiannaki, **Toward the Accurate Identification of Network Applications**, PAM 2005
- M. de Vivo, et al., **A Review of Port Scanning Techniques**, ACM CCR, 1999
- **Extensible Markup Language (XML)**, http://www.w3c.org/xml
- **XML-RPC**, http://www.xmlrpc.com/