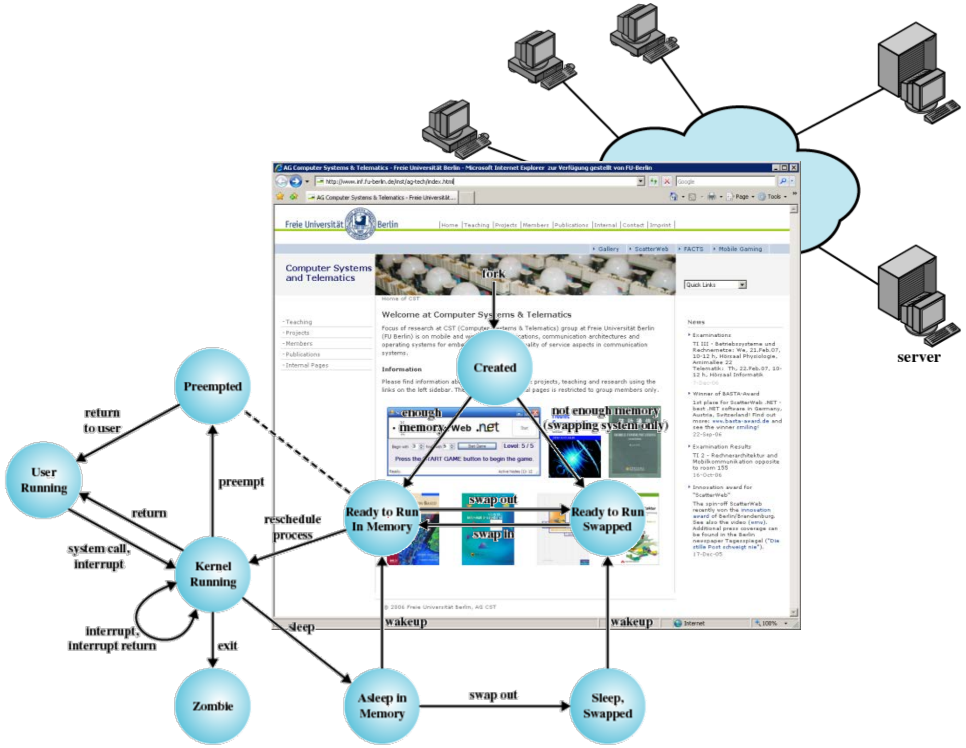
TI III: Operating Systems & Computer Networks
Example

Prof. Dr.-Ing. Jochen Schiller
Computer Systems & Telematics
Freie Universität Berlin, Germany



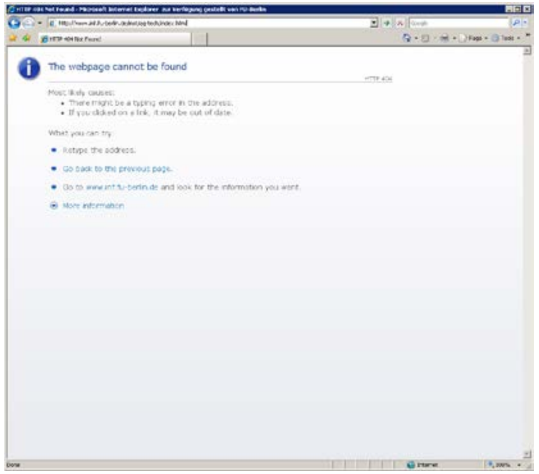
Content

A Comprehensive Example

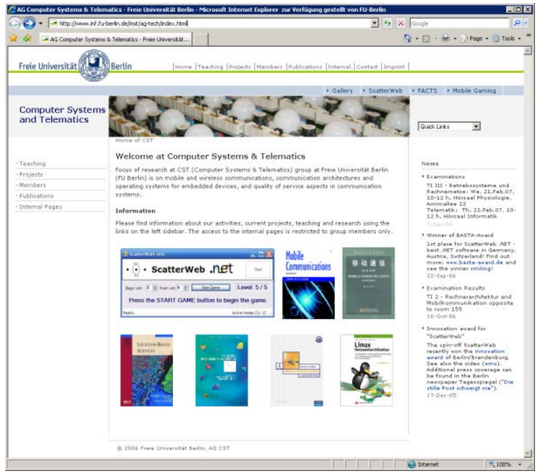
What happens if one presses a key on the computer?

?

What if that key causes an web page to be displayed?



?

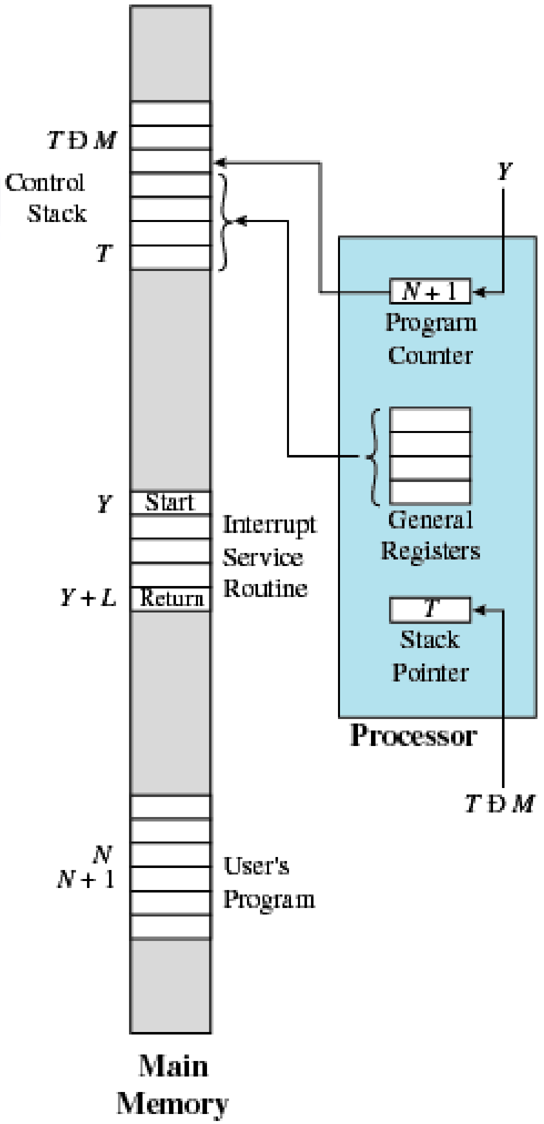


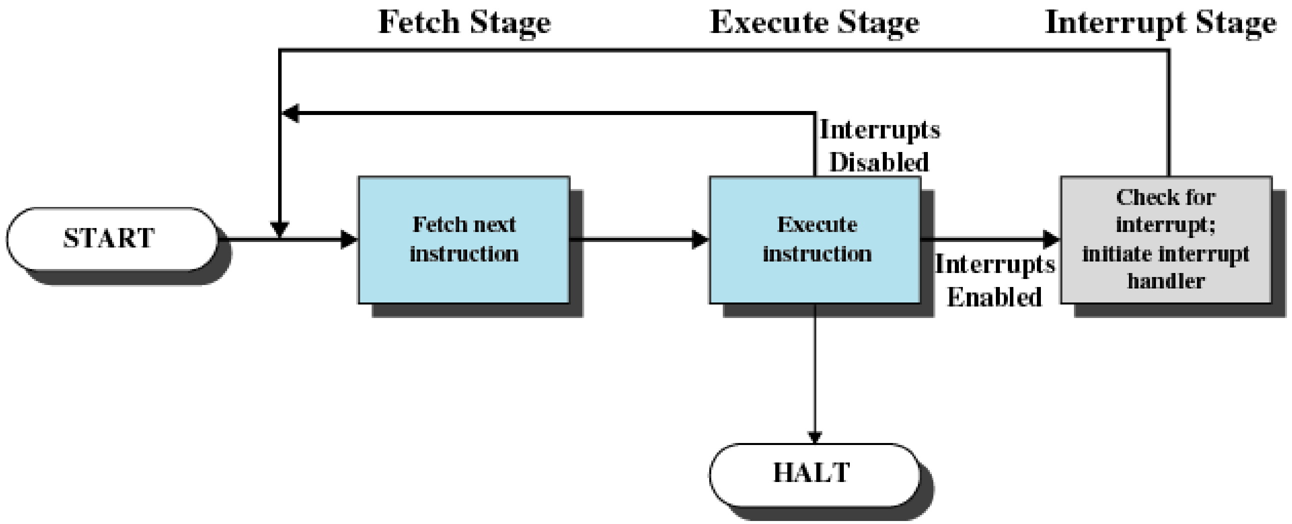
Keyboard Interrupt

Keyboard controller raises interrupt flag

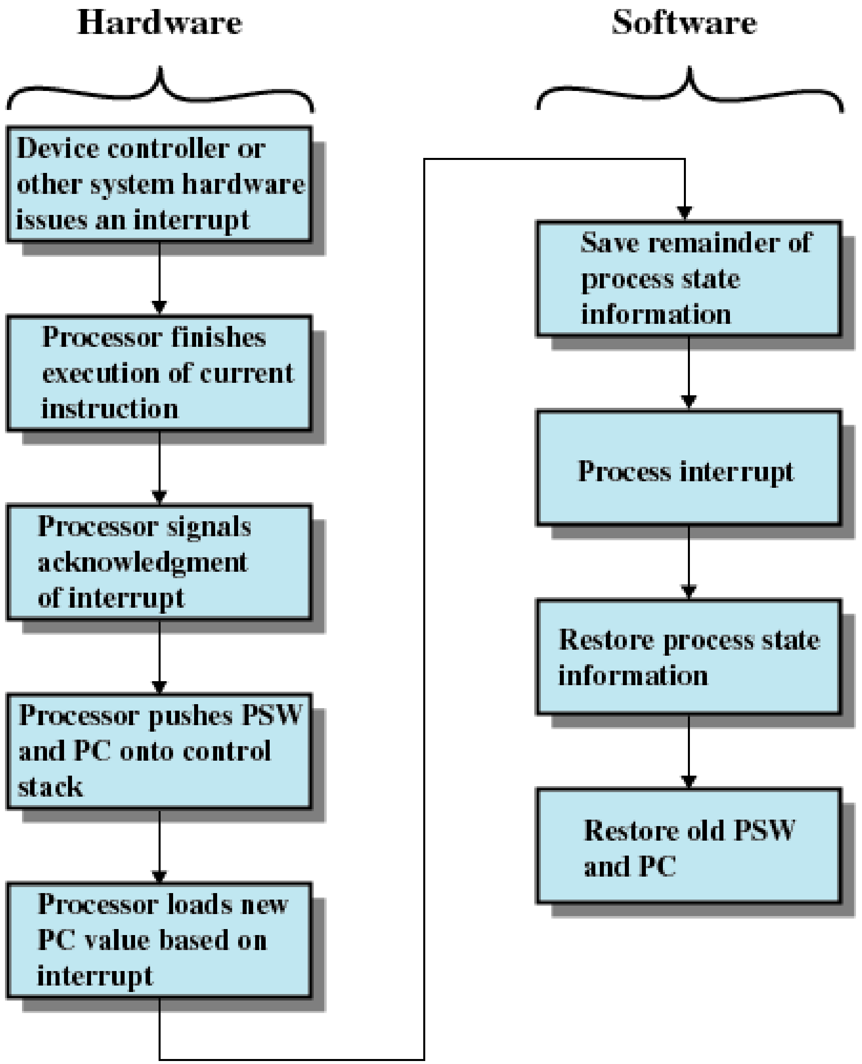
CPU interrupts execution of current process and starts Interrupt Service Routine (ISR)

- Unconditional jump



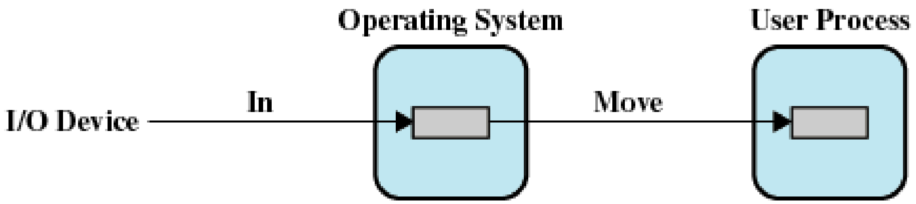


Keyboard Interrupt Handling



ISR processes input from keyboard

- Clears interrupt flag
- Transfers data from device into buffer
- Establishes owner of device
- Triggers notification of user process

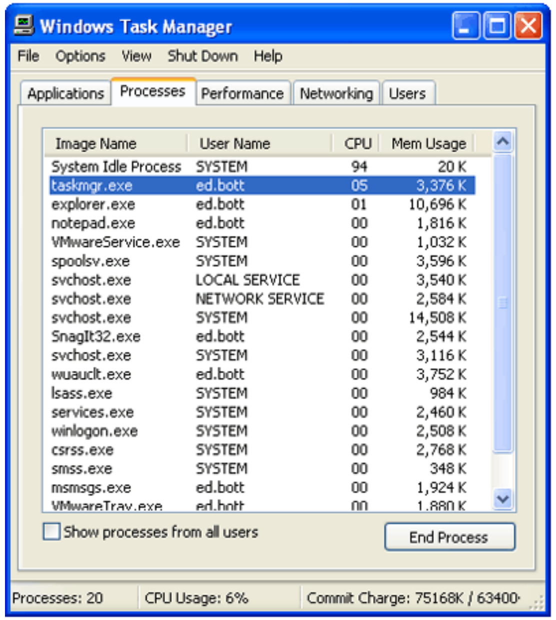


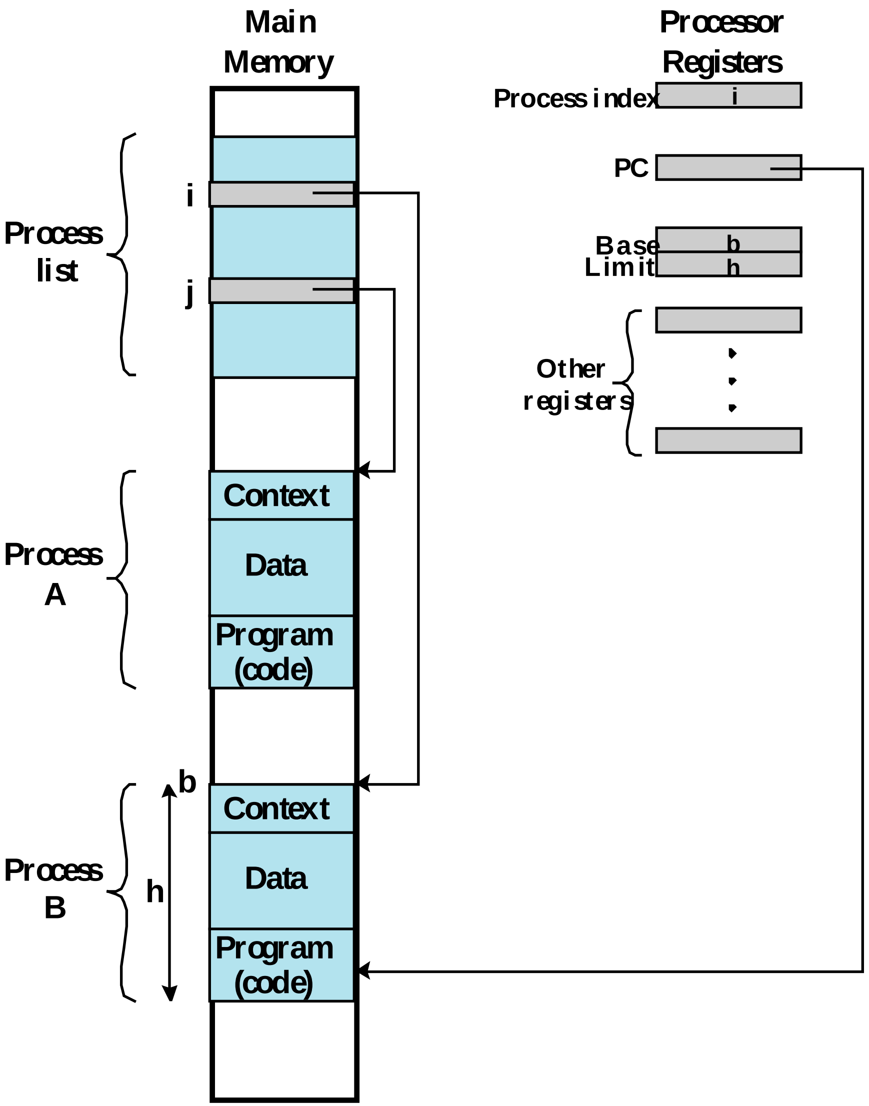
In the Meantime…

Web browser is one of many processes running locally

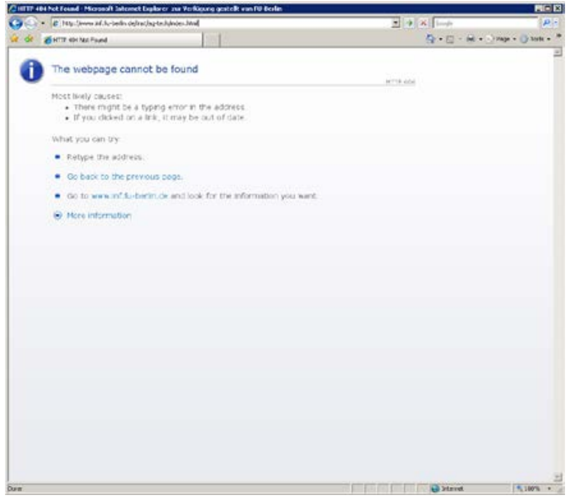
Other processes include

- Other user processes (possibly of different users)
- System processes implementing system services
- Kernel processes



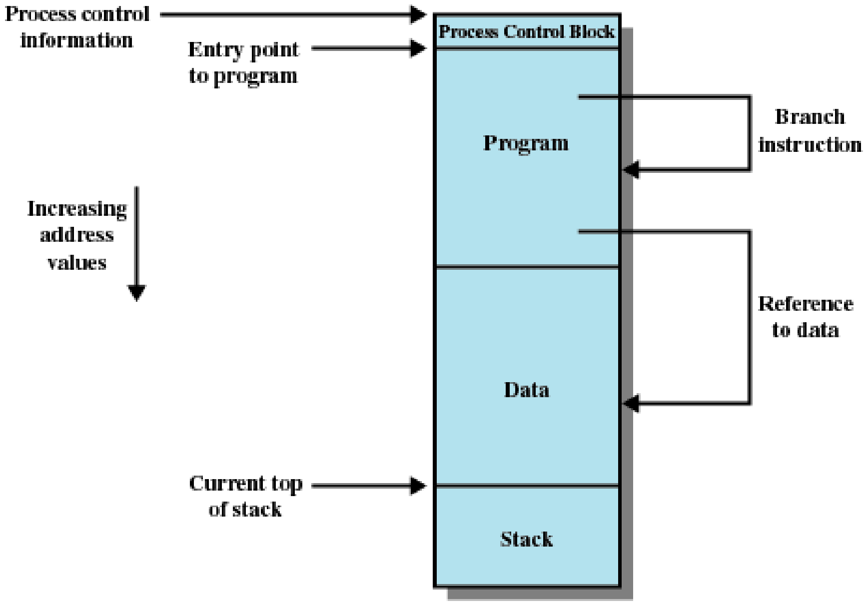


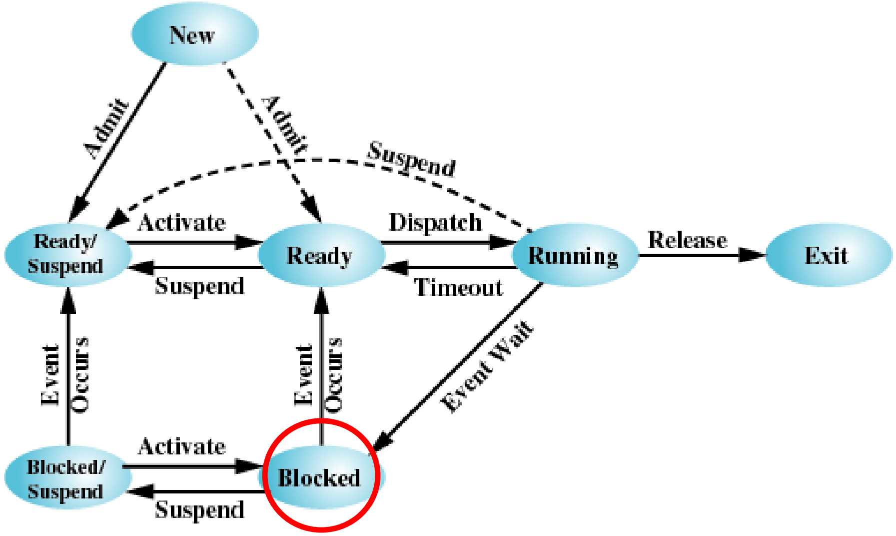
Web Browser Process in Detail



Web browser processes

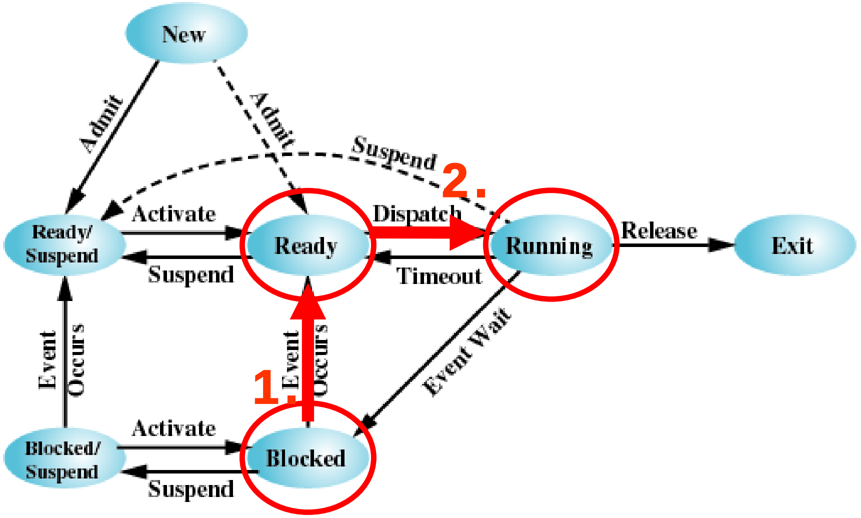
- ➢Currently waiting for input
  - E.g. using **`select()`**
- ➢Process state **blocked**

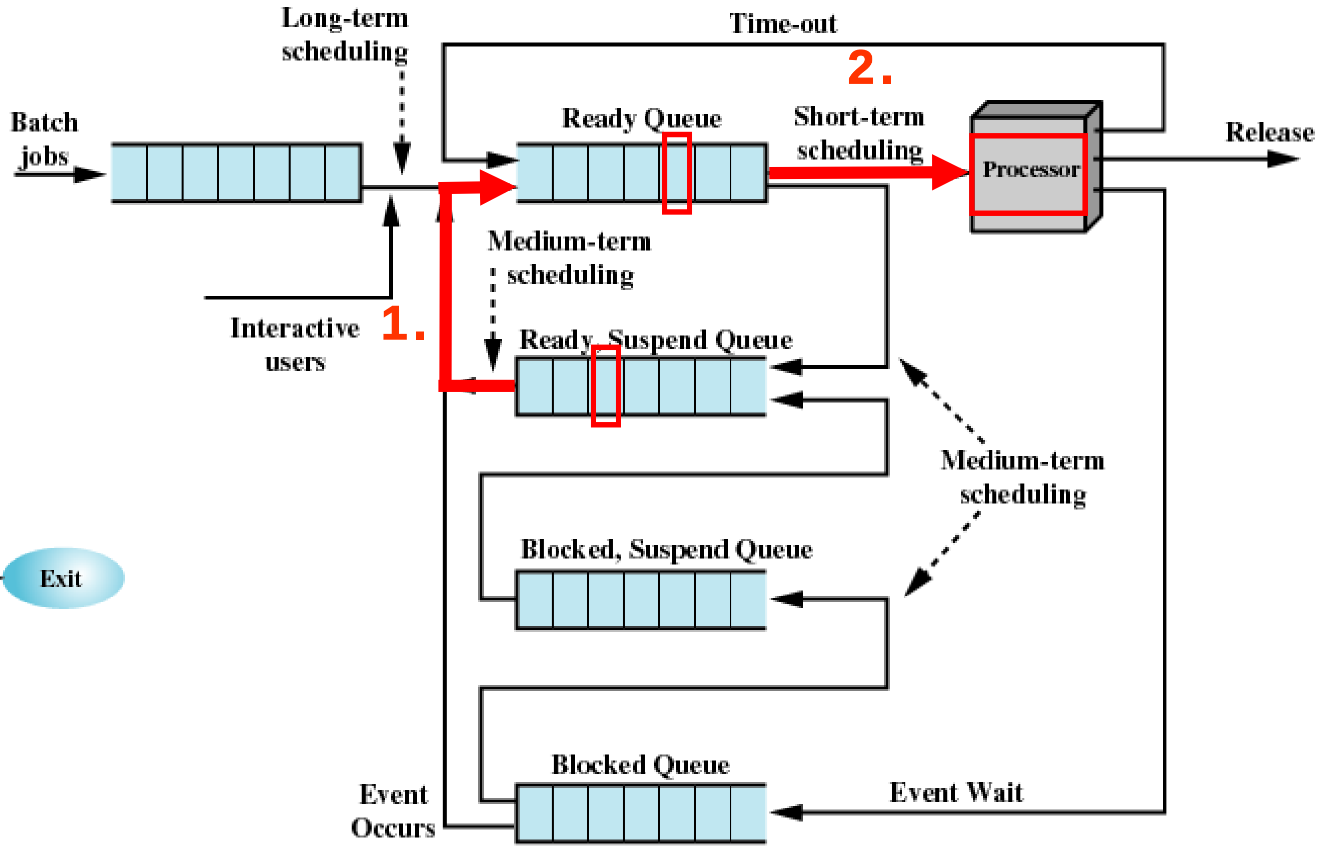




Reaction to External Event

1. ISR changes process state to **ready**
2. Scheduling algorithm eventually changes process state to **running**





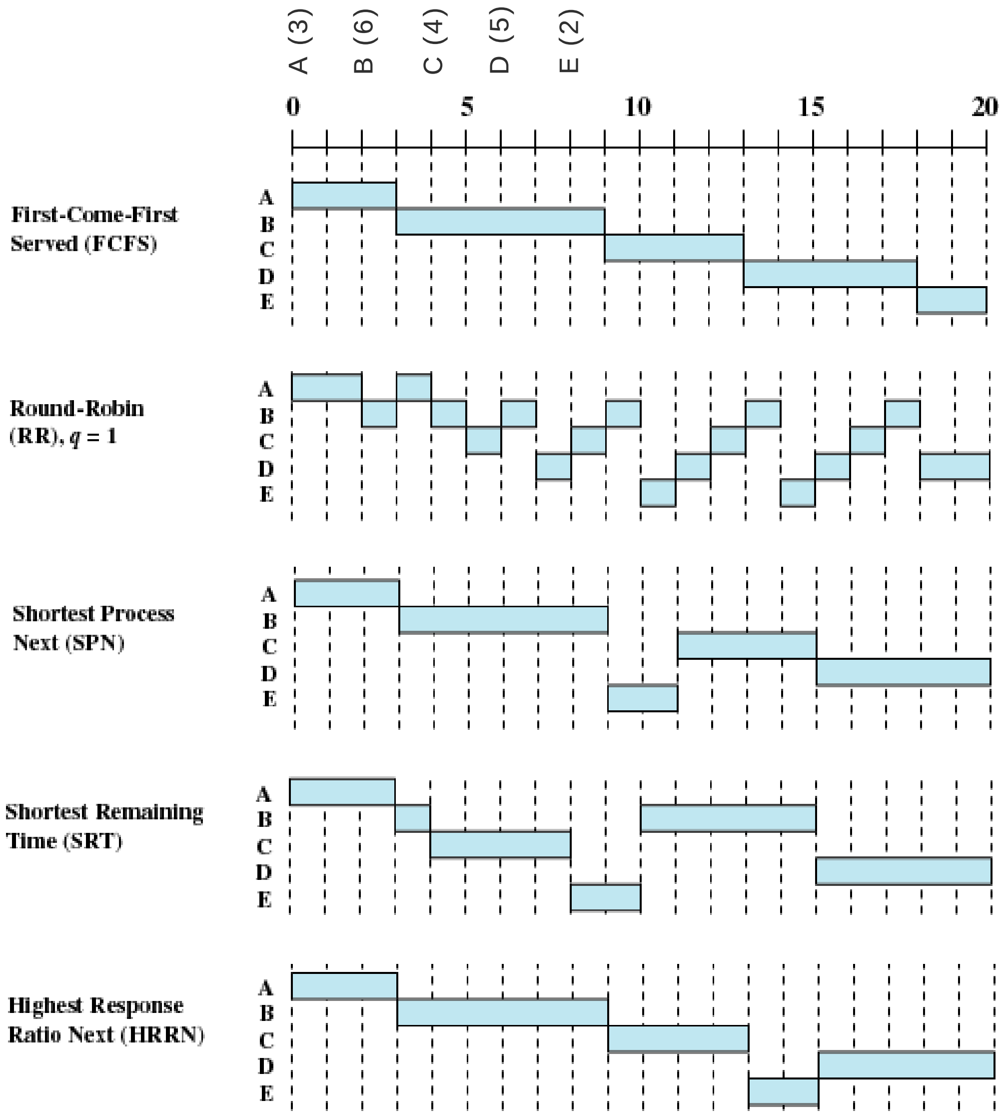
Process Scheduling

Scheduling is handled by variety of scheduling algorithms

- Non-preemptive / preemptive
- Maximize throughput, responsiveness, etc...

Processes may have priorities

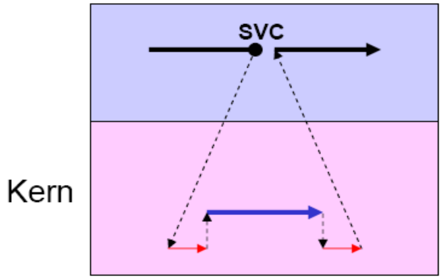
- Priority inversion due to lock on shared resources
- Priority inheritance



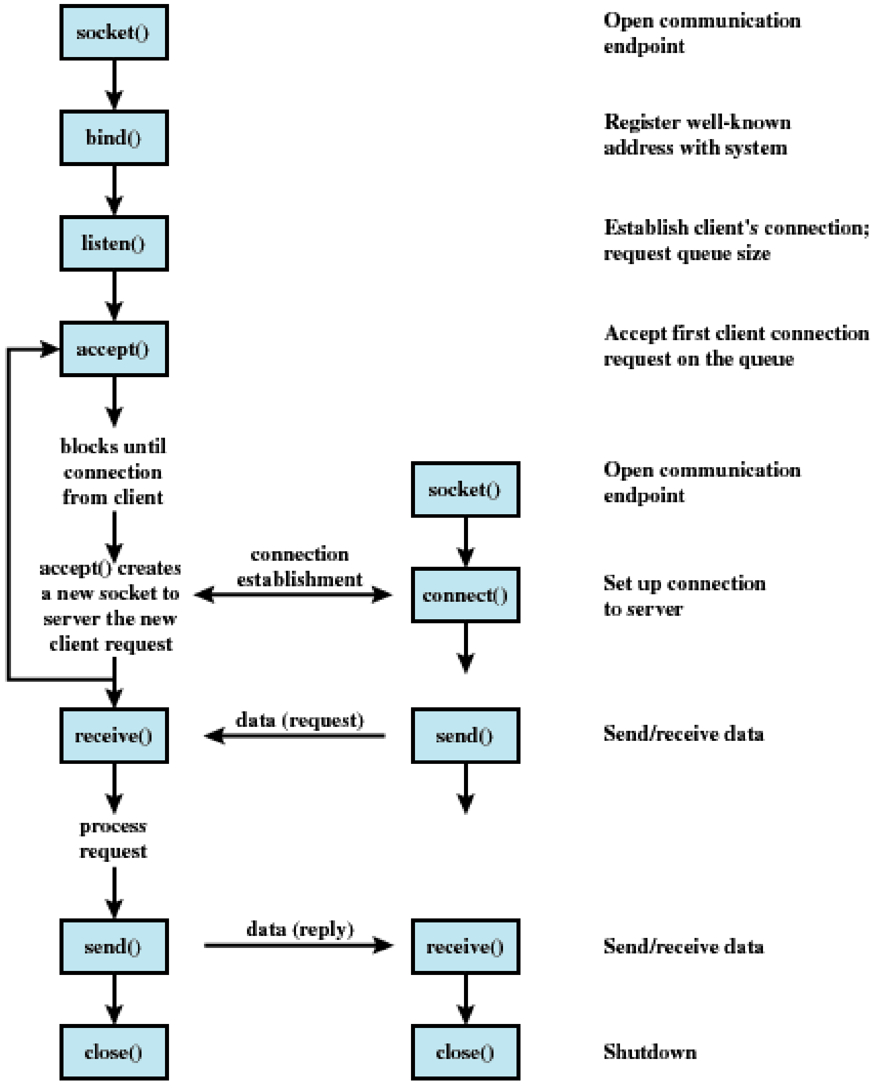
Web Browser Processes Event

Assume input requires web browser to display a web page with a given URL

1. String processing (user space)
2. Connect to server and retrieve necessary data (system calls)

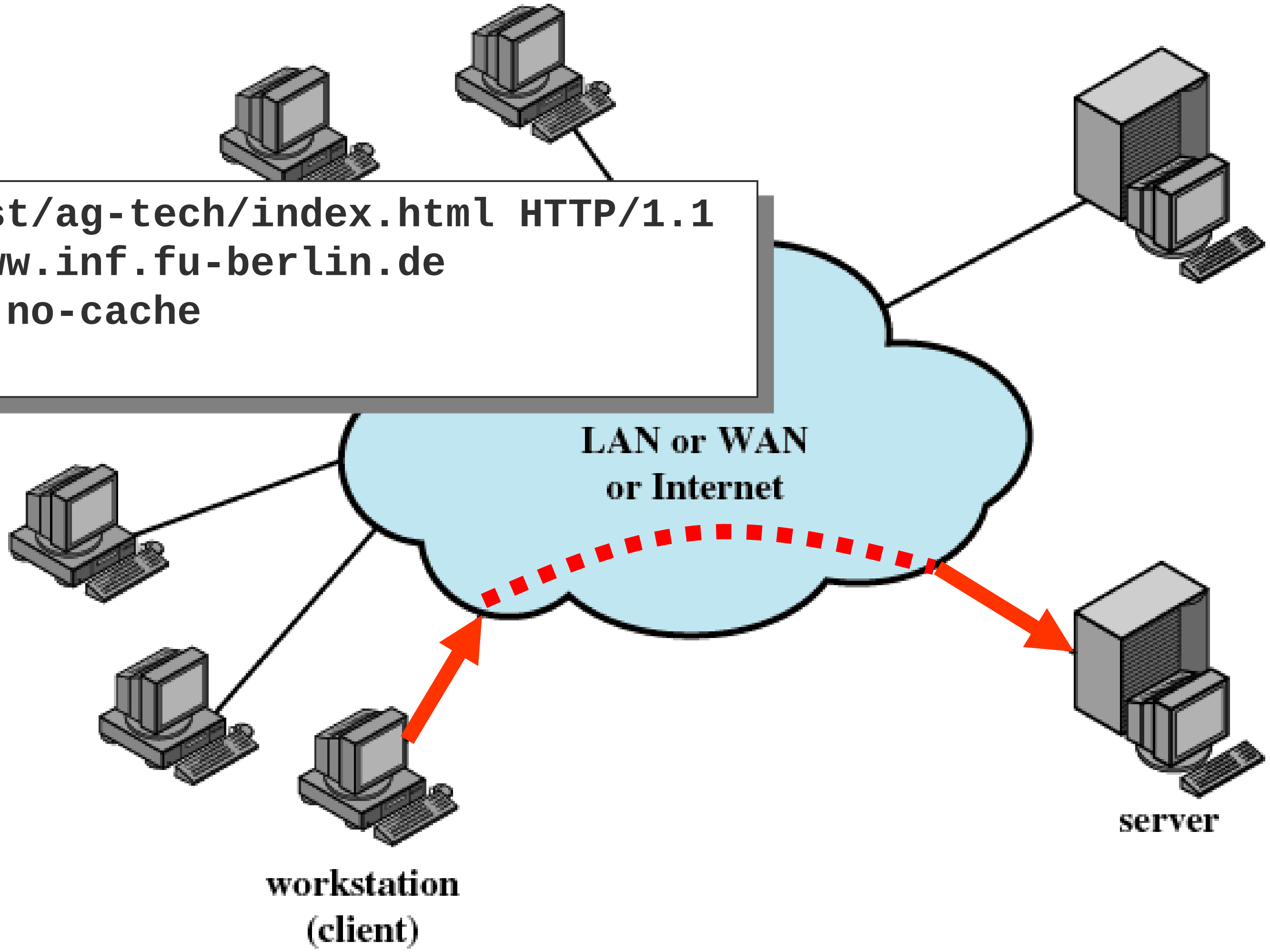


3. Render web page (user space)
4. Update user interface (system calls)

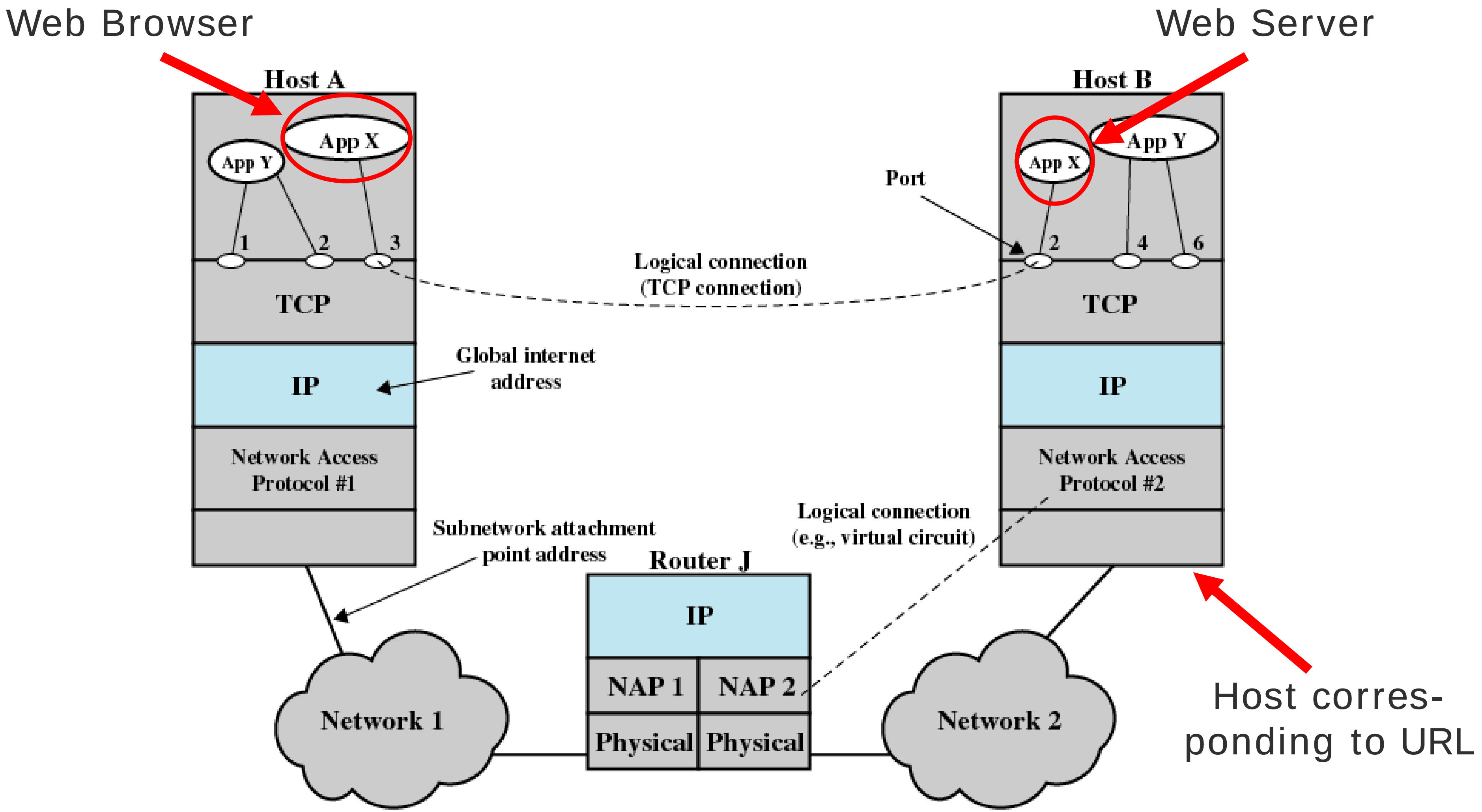


Client/Server Communication

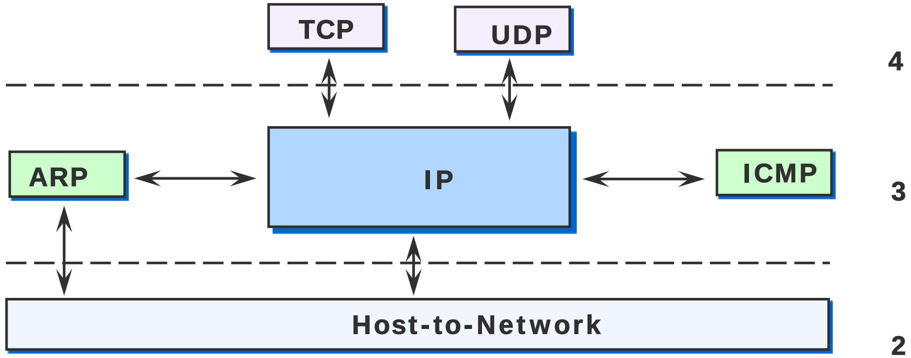
```
GET /inst/ag-tech/index.html HTTP/1.1
Host: www.inf.fu-berlin.de
Pragma: no-cache
....
```



Layered Protocol Stack



Interaction Between Network Layers

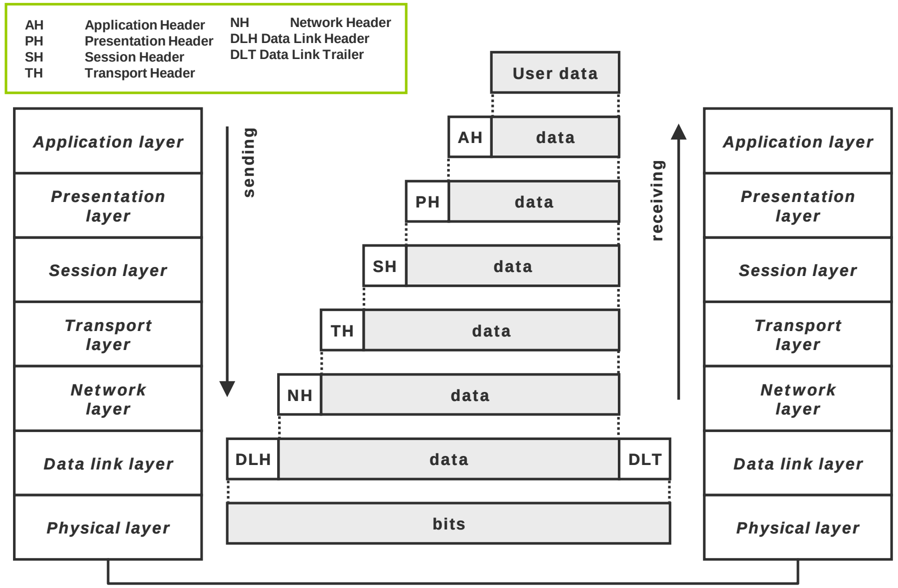


Layered protocol architecture

- Each layer uses only services of layer directly below
- Each layer provides services to layer directly above

➢ Protocol independence

➢ Modularity



Data encapsulation

- Lower layers treat upper layer packets as simple data
- Headers contain control information for each layer

➢ Repeated encapsulation causes overhead

Uniform Resource Locator (URL)

`http://cst.mi.fu-berlin.de/index.html`

`http`: Hypertext Transfer Protocol (HTTP)

- Protocol for accessing web pages and related content
- Implies communication over port 80 (unless other port given in URL)

`cst.mi.fu-berlin.de`: Host name

- Resolved to IP address via Domain Name System (DNS)
- **`cst.mi.fu-berlin.de`** -> 160.45.117.167

`index.html`: Local resource name

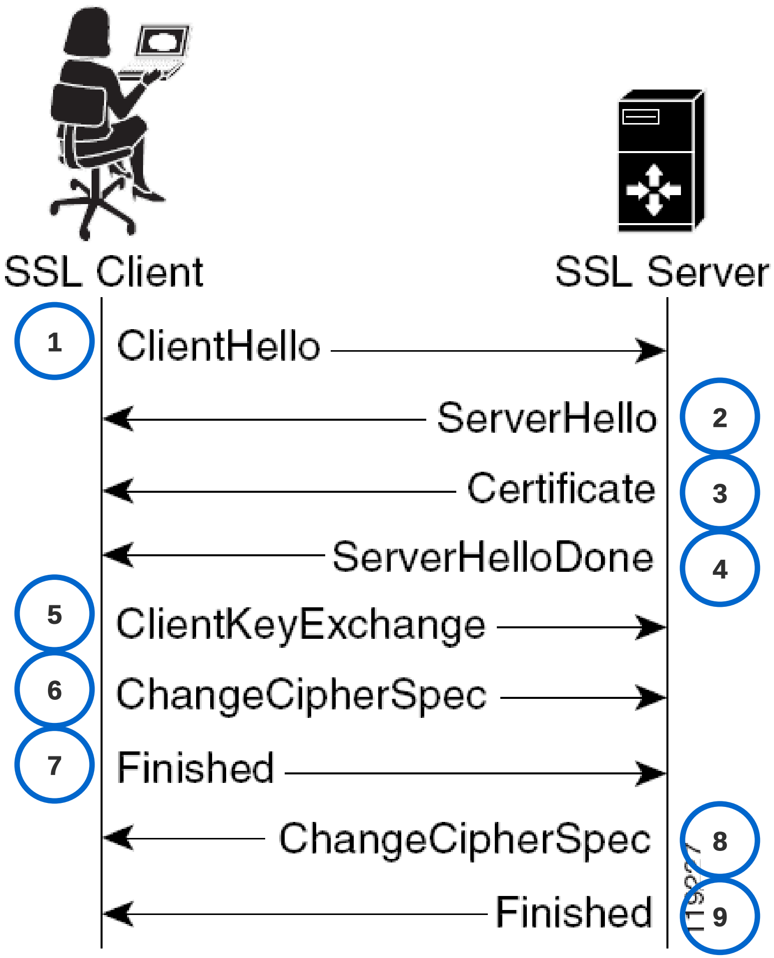
- Protocol specific parameter
- Handled by web server

Security: HTTP over TLS/SSL

HTTPS authenticates server and establishes secure connection:

1) Propose SSL parameters, send random number
2) Agree to parameters, send random number
3) Send public key certificate
4) Conclude handshake negotiation
5) Send random number encrypted with server's public key
   - Client and server derive session key from all three random numbers
6) Activate negotiated parameters
7) Send encrypted hash over previous messages
   - Server decrypts and verifies message
8) Activate negotiated parameters
9) Send encrypted hash over previous messages
   - Client decrypts and verifies message

➢ Proceed to exchange regular HTTP data over secure channel



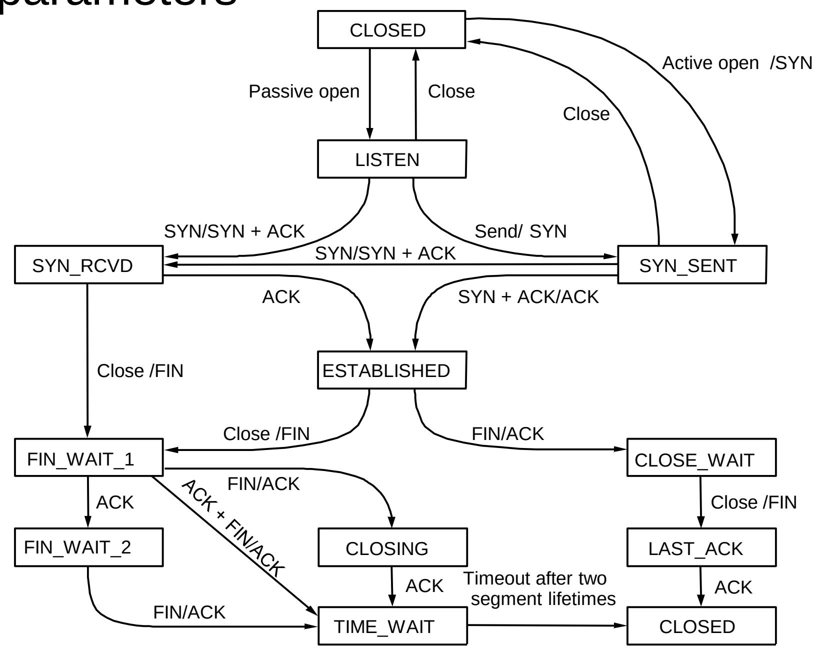
Source: Cisco Systems. Application Control Engine Module SSL Configuration Guide

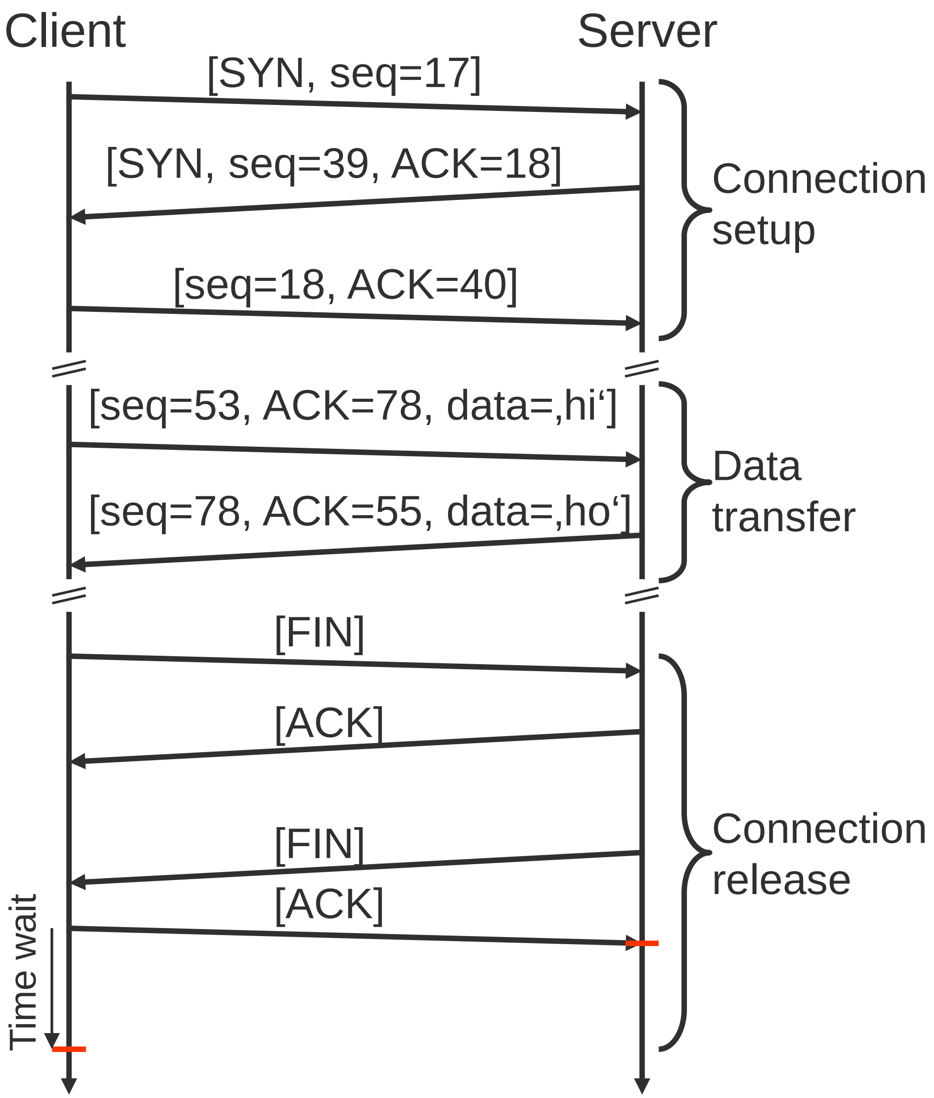
Connection Setup / Transport Layer

Reliable end-to-end connection between processes

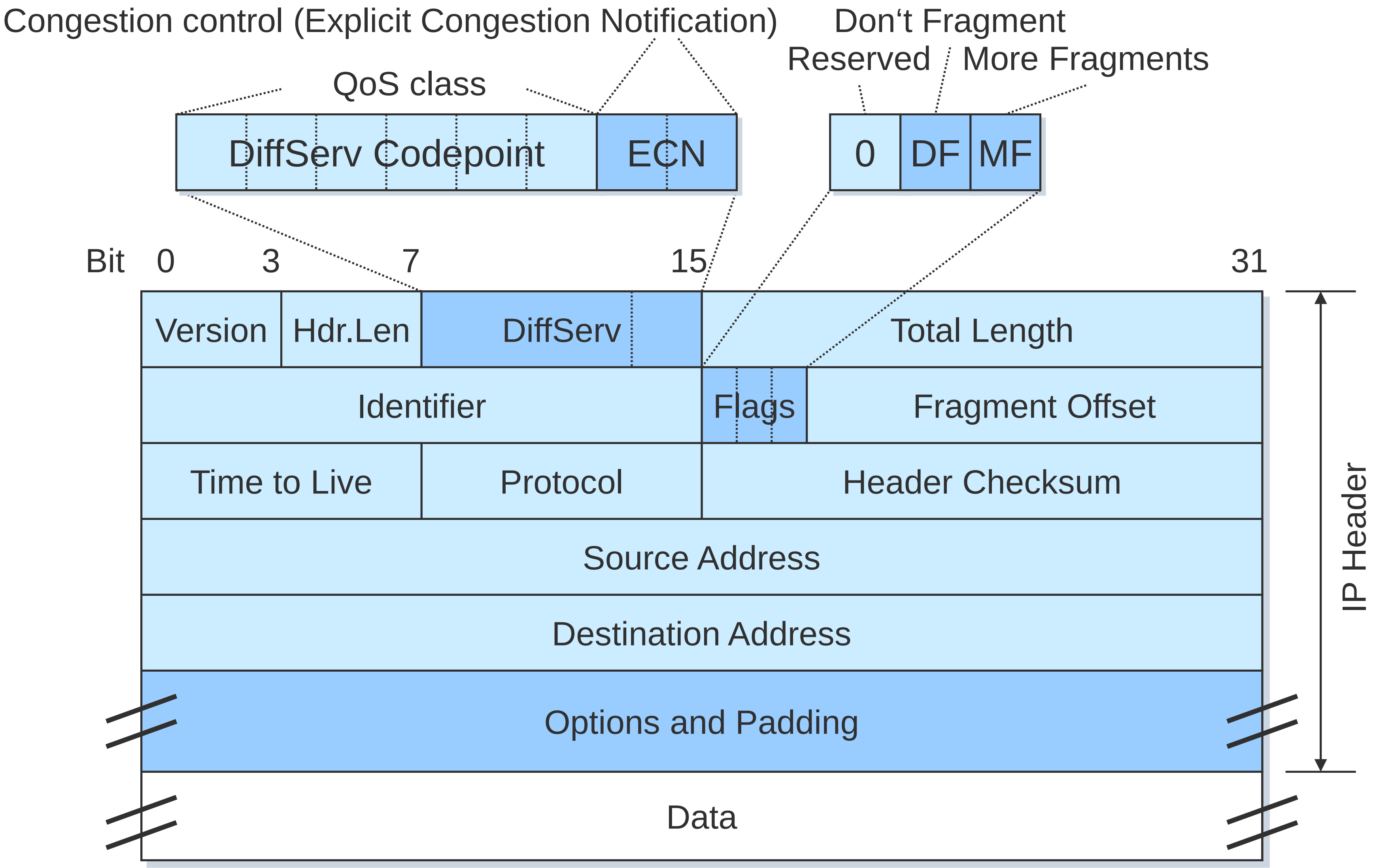
Call to **`connect()`** initiates connection setup

- TCP 3-way handshake
- Connection parameters





Structure of Network Layer IP-Packet



Network Layer Routing (Local Scope)

Globally unique per host addressing

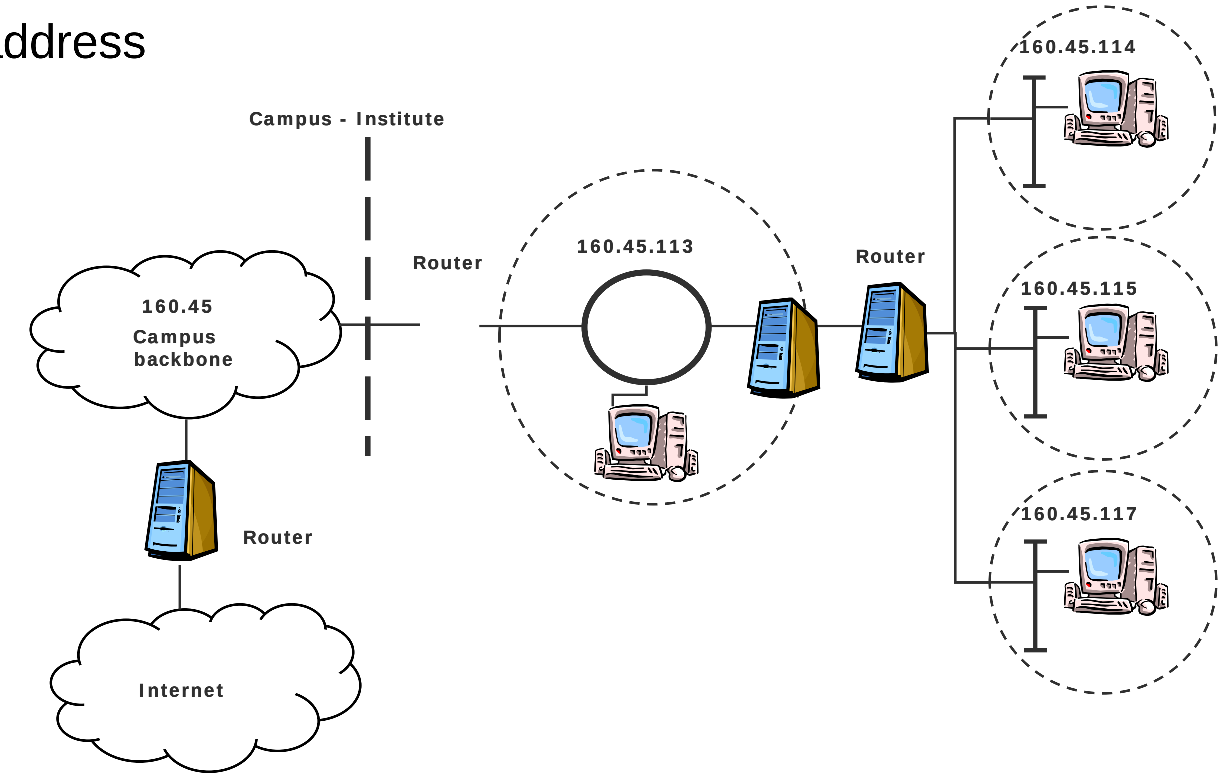
Routers maintain tables of known networks

- Optional route to default gateway

Subnetting implements logical structure

- Subnet mask builds hierarchy using host part of IP address
- Limits broadcasts
- More efficient routing

Network topology may be part of security concept

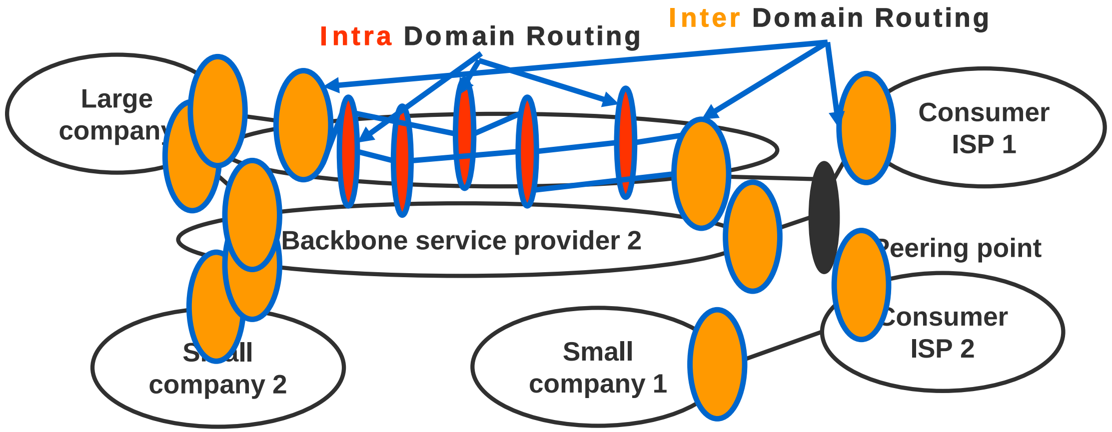


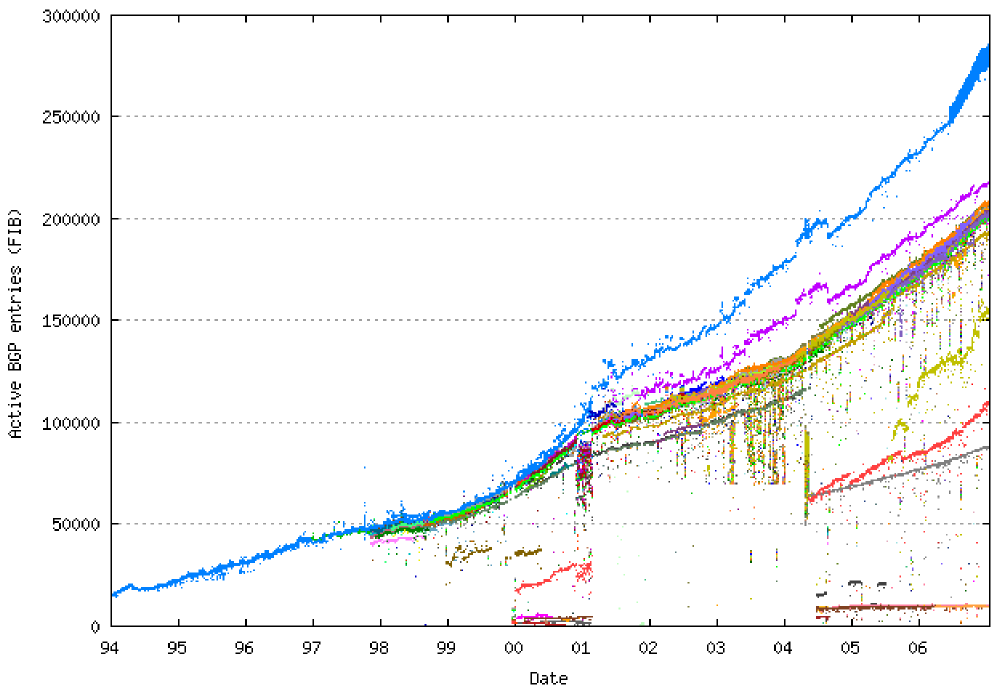
Network Layer Routing (Global Scope)

Internet organized into autonomous systems (AS)
- Commonly, one AS per major organization
- Peering points to exchange data between ASs

Intra-domain routing: OSPF, link state algorithm

Inter-domain routing: BGPv4, distance vector protocol
- May involve non-technical routing choices

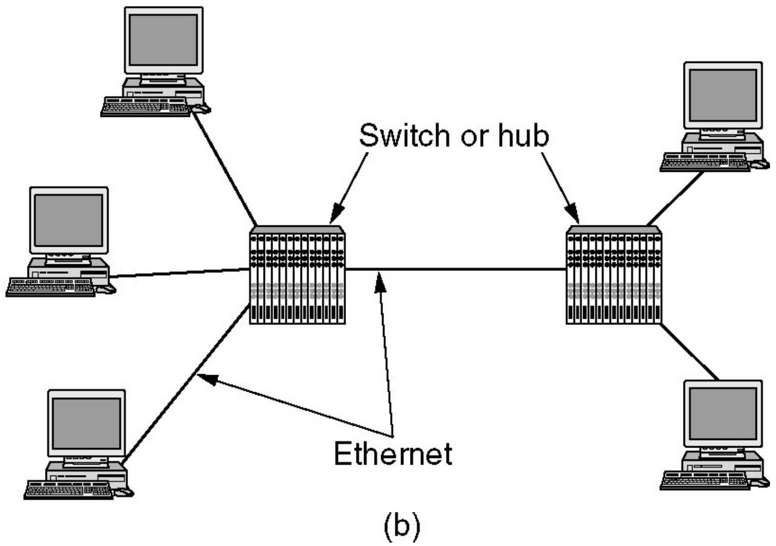


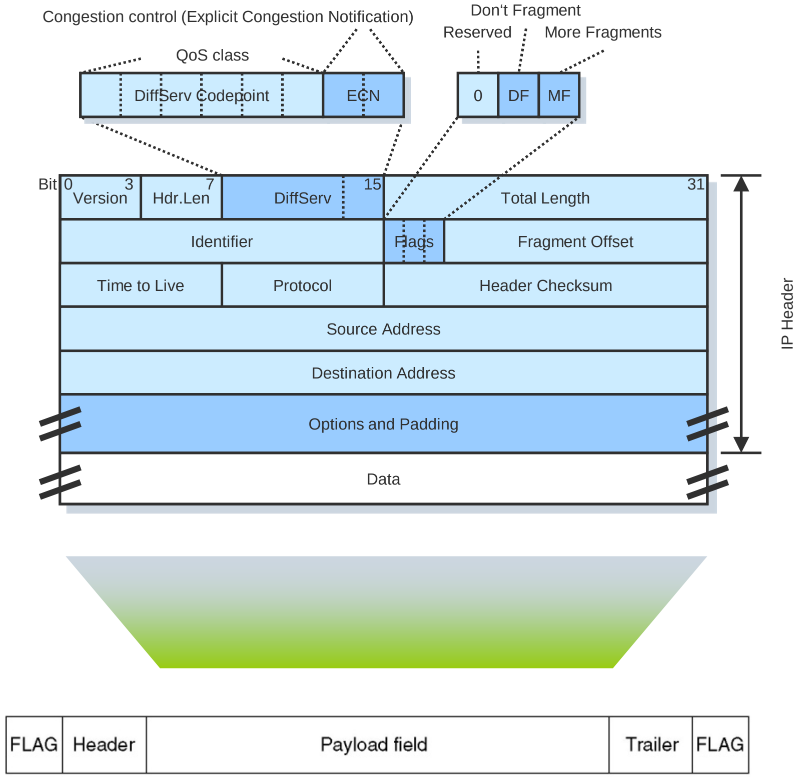


Data Link Layer Communication (Local Scope)

Transparent communication between two directly connected nodes

Services include: framing, error control, connection maintenance, acknowledgements, flow control





Error Detection: Cyclic Redundancy Check (CRC)

- Reception of a correct bit sequence:

```
11 0011 1001 ÷ 1 1001 = 10 0001 (mod 2)
11 001
00 0001 1001
        1 1001
        0 0000  = remainder
```

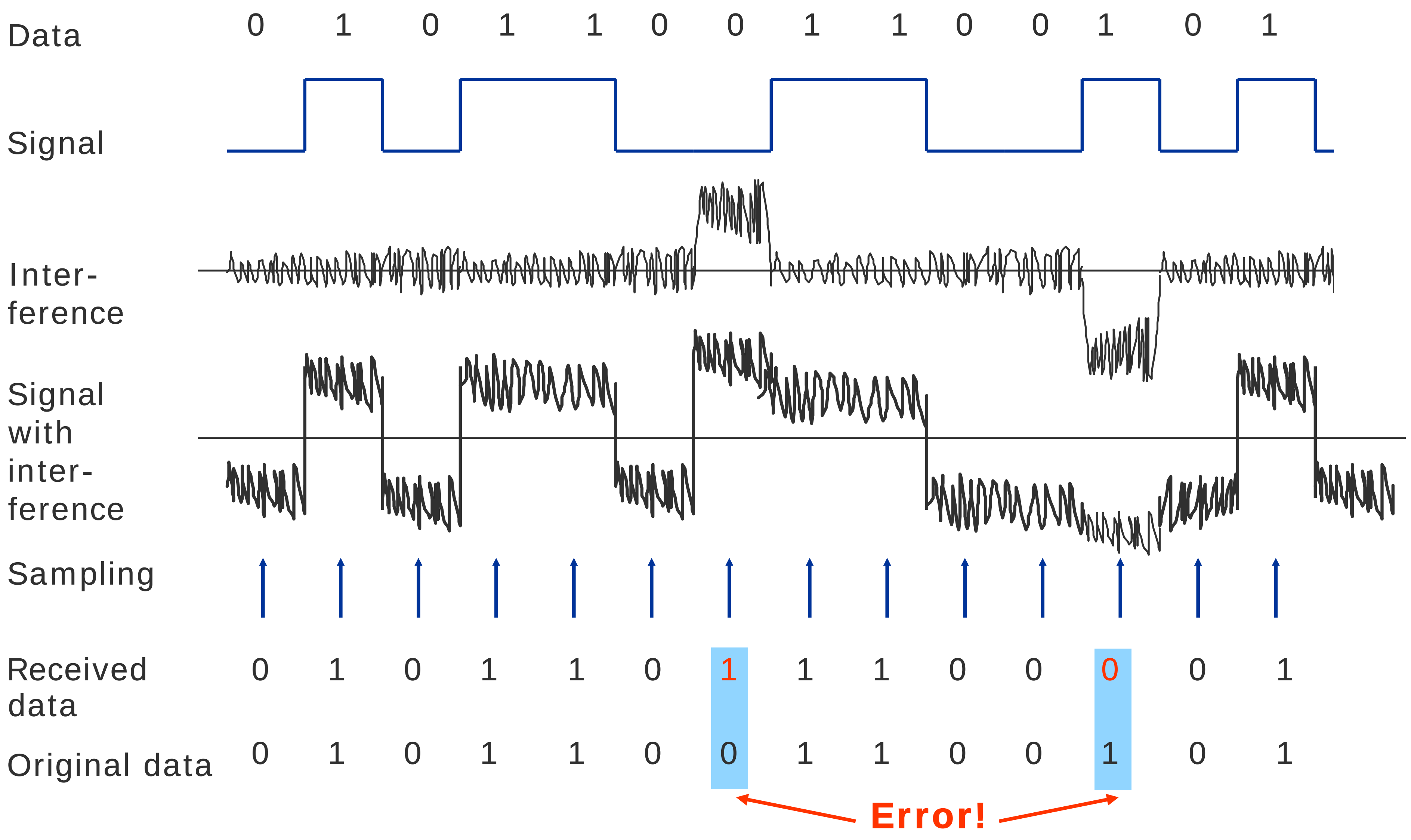
  - No remainder, thus the received bits *should* be error free

- Reception of a erroneous bit sequence:

```
11 1111 1000 ÷ 1 1001 = 10 1001 (mod 2)
11 001
00 1101 1
   1100 1
   0001 0000
        1 1001
        0 1001  = remainder ≠ 0
```

  - There is a remainder unequal 0, thus there was definitely a transmission error

Errors During Transmission



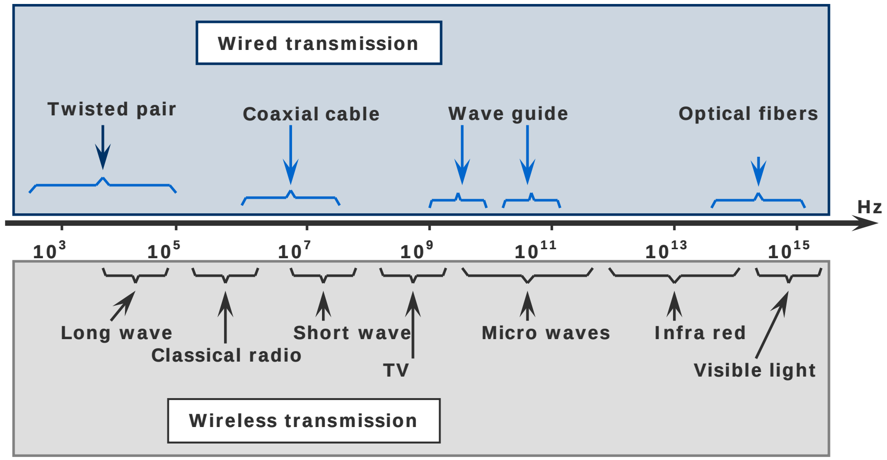
Physical Layer

Packet / sequence of bits turned into physical signal

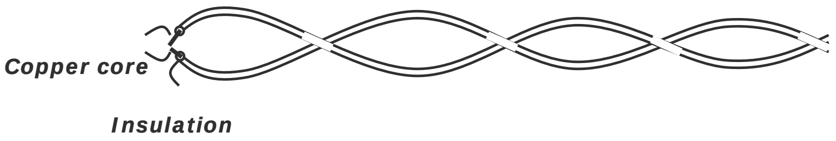
Signal propagation depends on physical medium (limited bandwidth, attenuation, dispersion) and background noise

Mapping between bits and (multi-valued) symbols

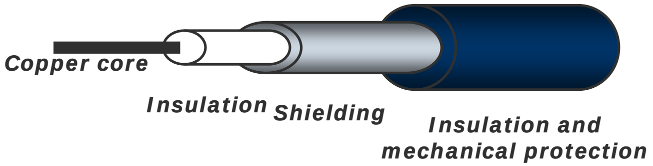
Baseband transmission vs. modulation (broadband transmission)



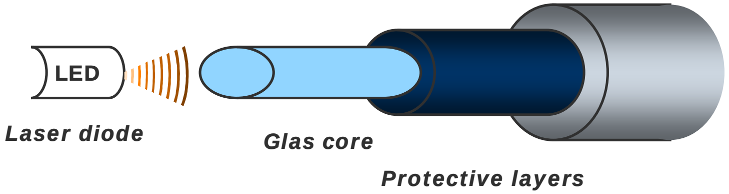
Twisted pair



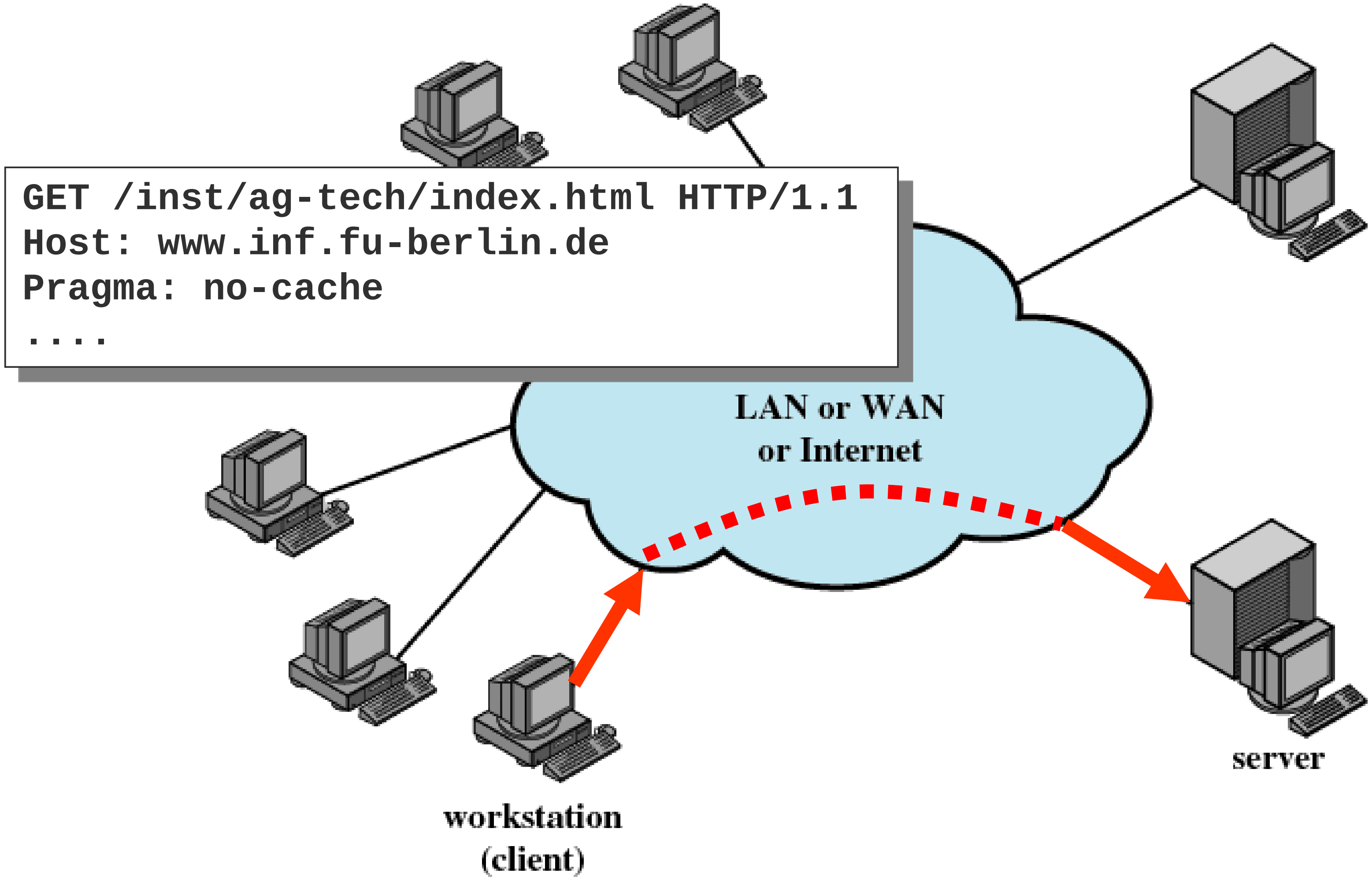
Coaxial



Optical fiber

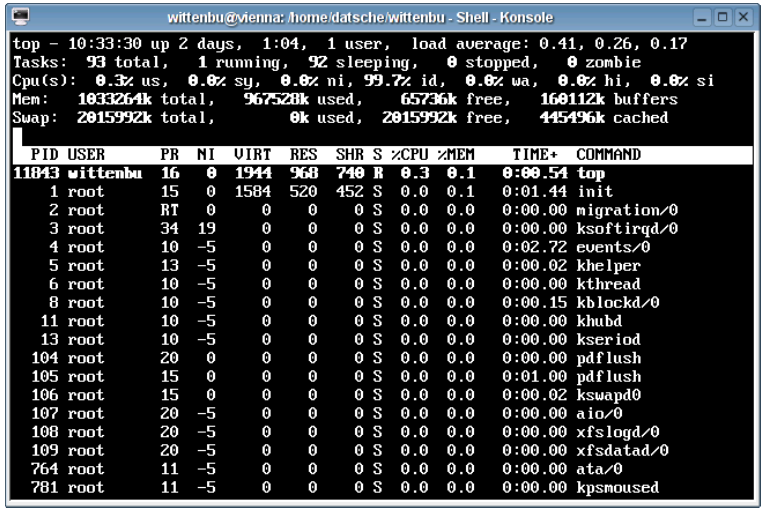


Client/Server Communication



At the Server…

Web server is one of many processes running locally



```
wittenbu@vienna: /home/datsche/wittenbu - Shell - Konsole
top - 10:33:30 up 2 days,  1:04,  1 user,  load average: 0.41, 0.26, 0.17
Tasks:  93 total,   1 running,  92 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.3% us,  0.0% sy,  0.0% ni, 99.7% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:   1033264k total,   967528k used,    65736k free,   160112k buffers
Swap:  2015992k total,        0k used,  2015992k free,   445496k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
11843 wittenbu  16   0  1944  968  740 R  0.3  0.1   0:00.54 top
    1 root      15   0  1584  520  452 S  0.0  0.1   0:01.44 init
    2 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/0
    4 root      10  -5     0    0    0 S  0.0  0.0   0:02.72 events/0
    5 root      13  -5     0    0    0 S  0.0  0.0   0:00.02 khelper
    6 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 kthread
    8 root      10  -5     0    0    0 S  0.0  0.0   0:00.15 kblockd/0
   11 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 khubd
   13 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 kseriod
  104 root      20   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
  105 root      15   0     0    0    0 S  0.0  0.0   0:01.00 pdflush
  106 root      15   0     0    0    0 S  0.0  0.0   0:00.02 kswapd0
  107 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 aio/0
  108 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 xfslogd/0
  109 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 xfsdatad/0
  764 root      11  -5     0    0    0 S  0.0  0.0   0:00.00 ata/0
  781 root      11  -5     0    0    0 S  0.0  0.0   0:00.00 kpsmoused
```

- ➢Upon receiving packet, network interface controller (NIC) will raise interrupt
- ➢Kernel will handle the packet and notify the web server process

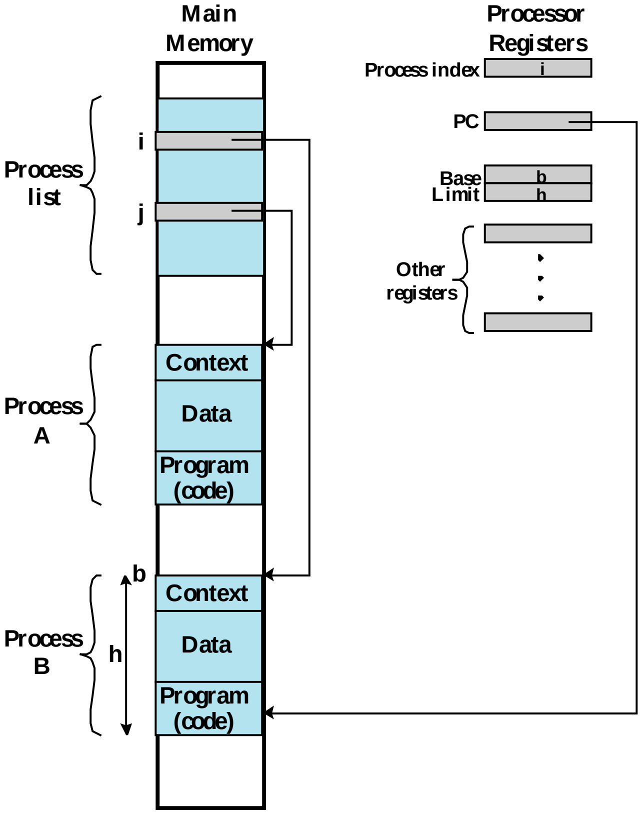


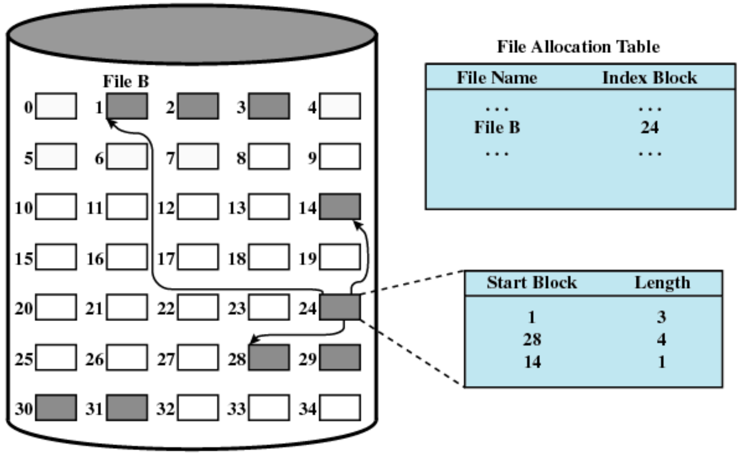
Figure 2.8 Typical Process Implementation

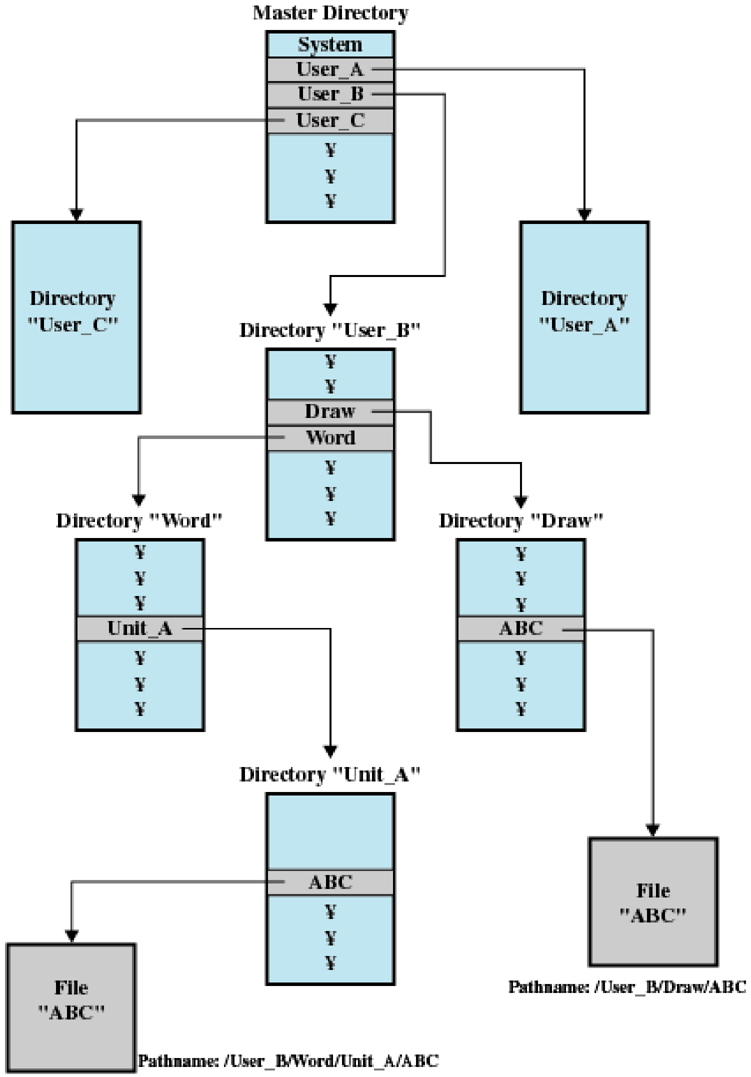
Processing of HTTP-GET Request

Web server retrieves file **`inst/ag-tech/index.html`** from local file system

- ➢System calls to access secondary storage
- ➢Kernel maps file name to data layout on disk

Web server sends data to client

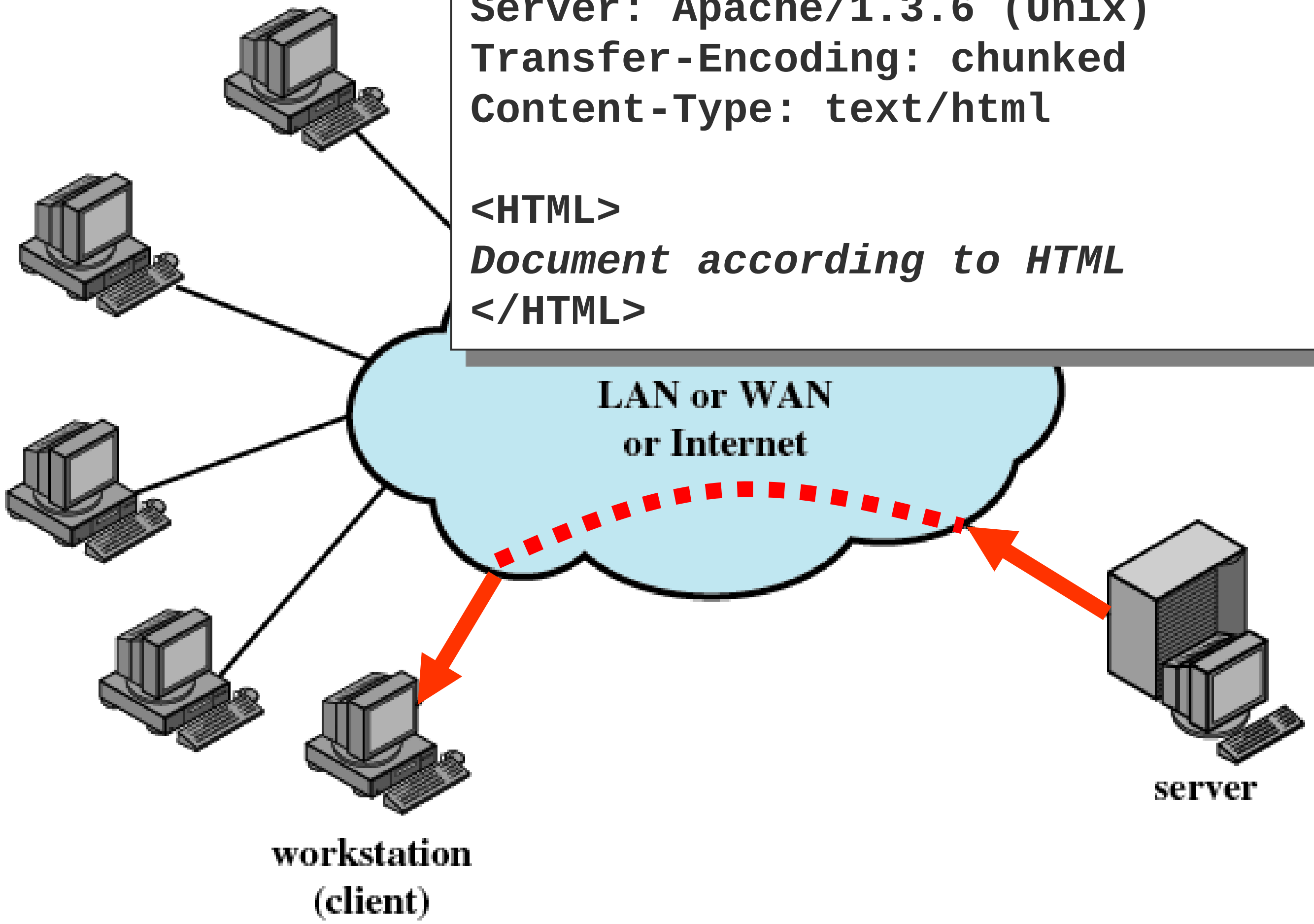




Server Replies to Client

```
HTTP/1.1 200 OK
Date: Fri, 16 Feb 2007 11:40:34 GMT
Server: Apache/1.3.6 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html

<HTML>
Document according to HTML
</HTML>
```



Client Data Processing

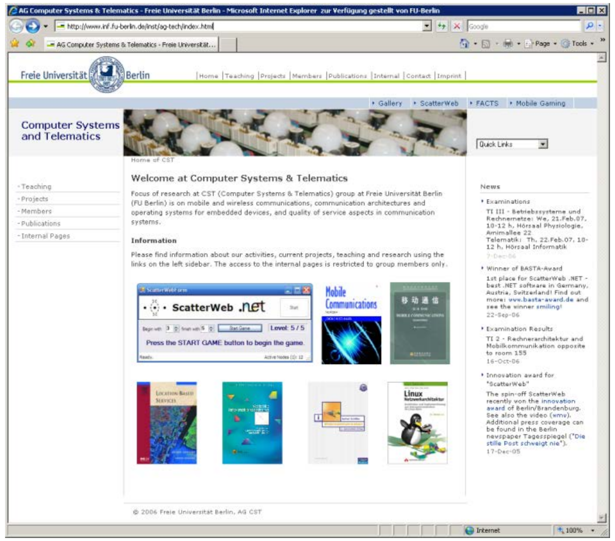
Client host receives packet

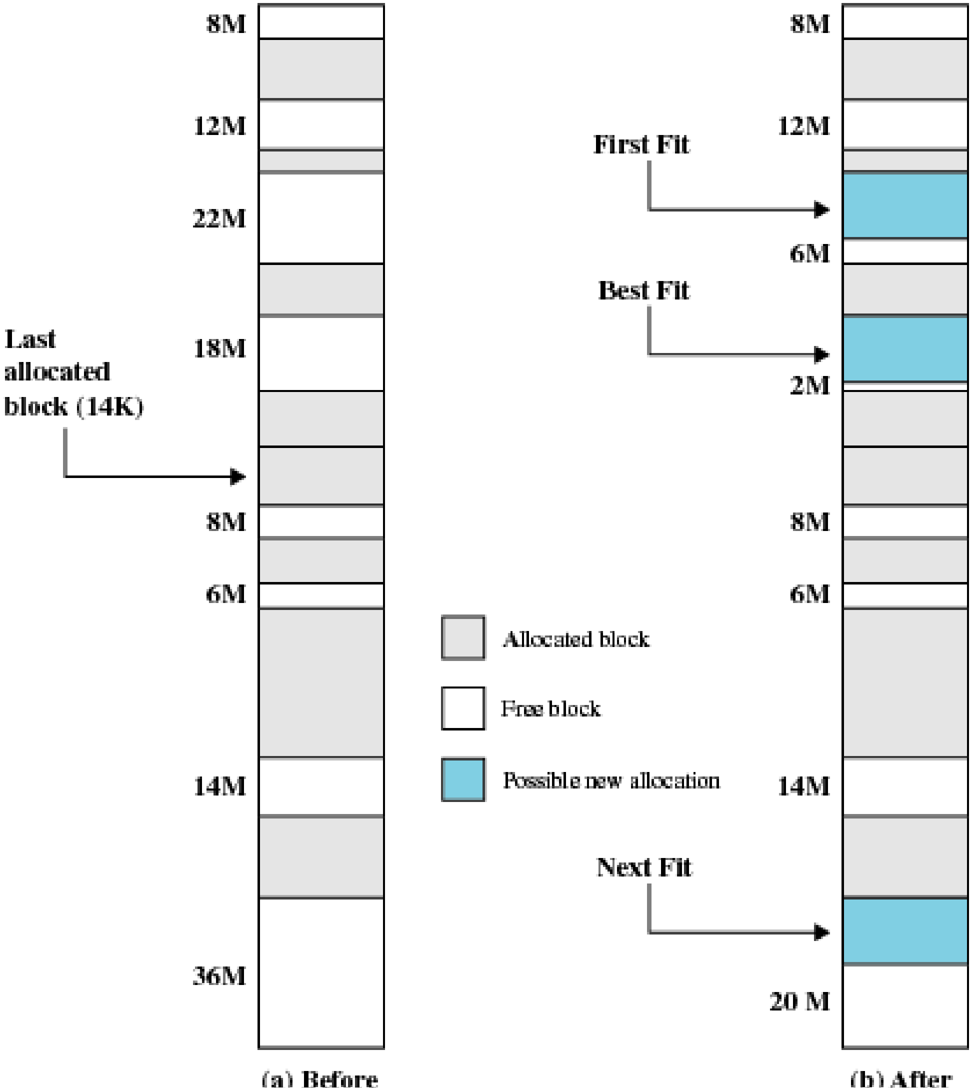
Kernel hands data to web browser process

Web browser renders page

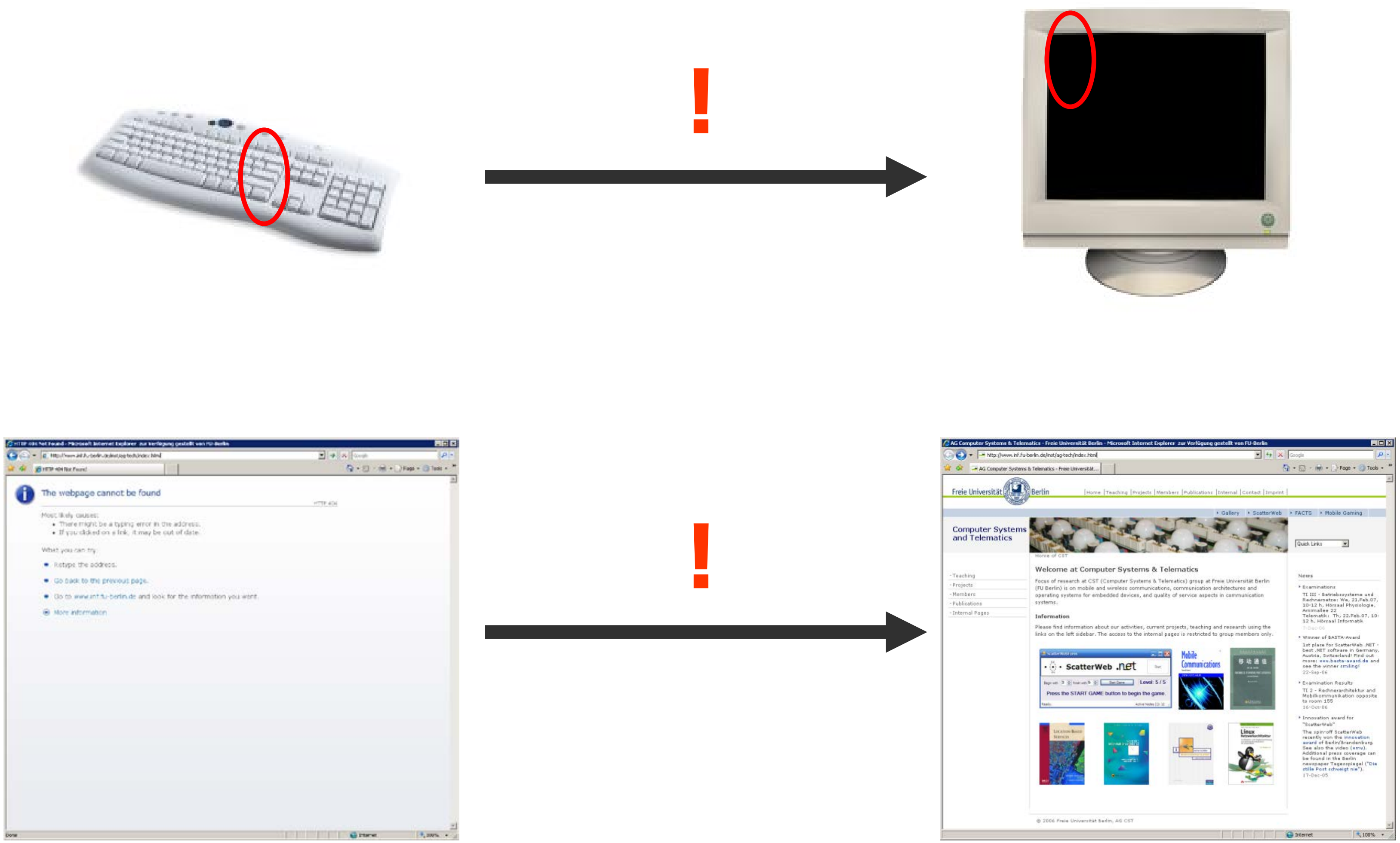
- May have to allocate memory in the process

Finally, browser updates user interface via system call





A Comprehensive Example



Content

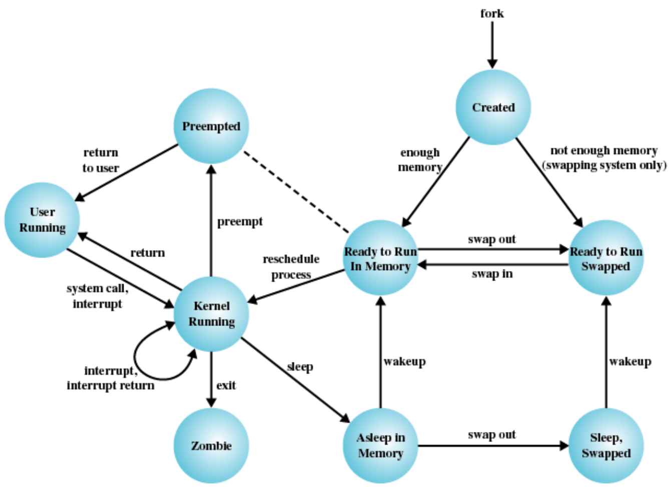
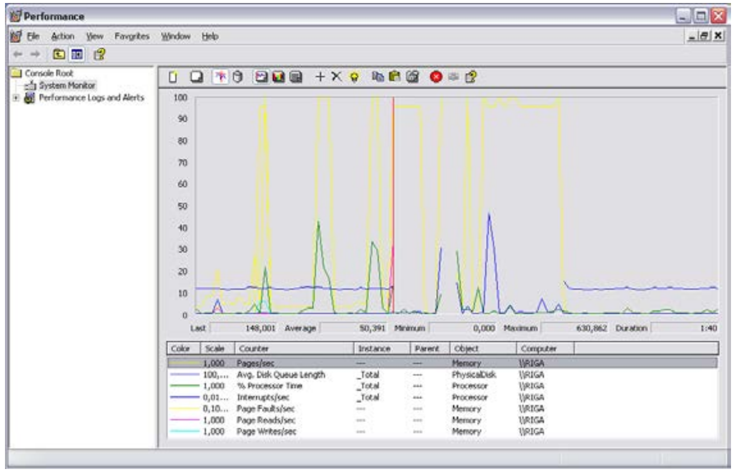


Figure 3.17 UNIX Process State Transition Diagram



Fin