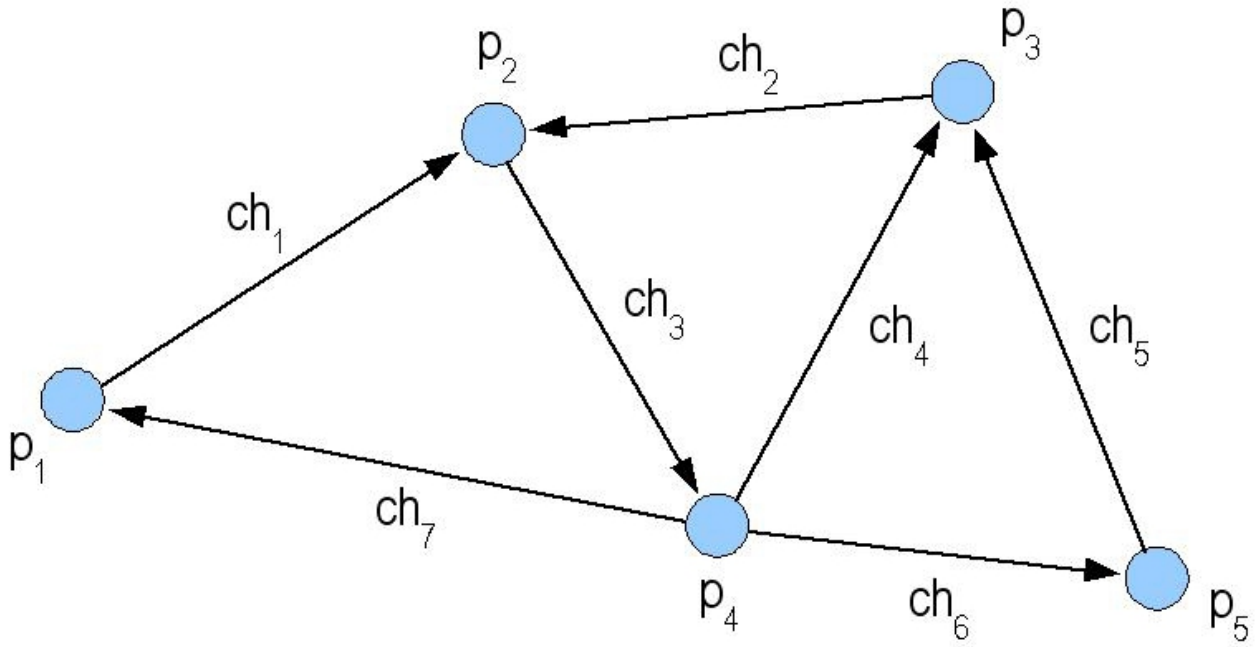
Structure of the MIE-DSV subject

areas (solved and close)

- parallel computing
- distributed computing
- client-server application
- clouds
- IoT - Internet of Things

Model of distributed computing

static system



G = (V,E)

V = { p_1, p_2, ... }

E = { ch_1, ch_2, ... }

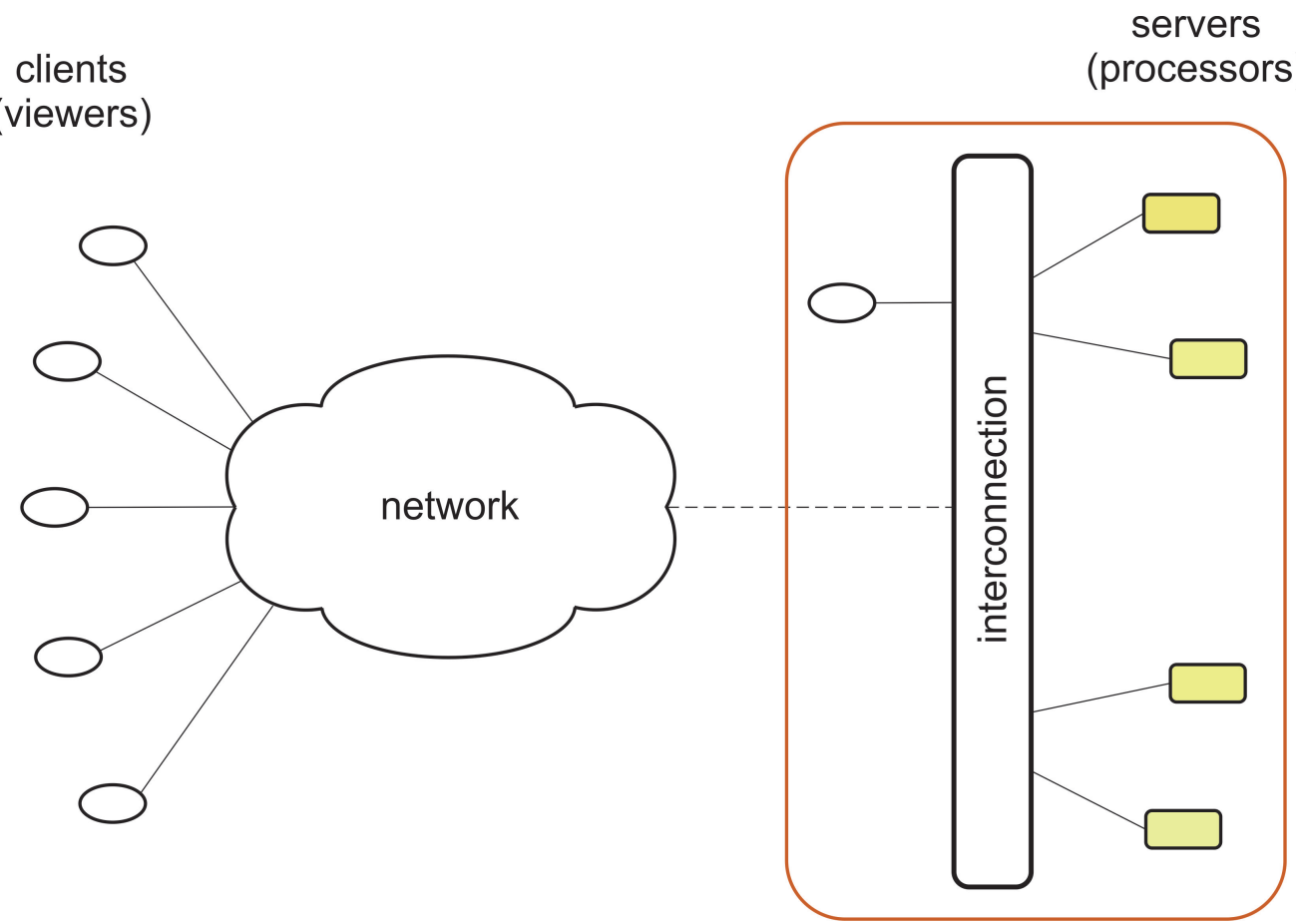
G - oriented graph

- processes
- communication among processes

Parallel computing

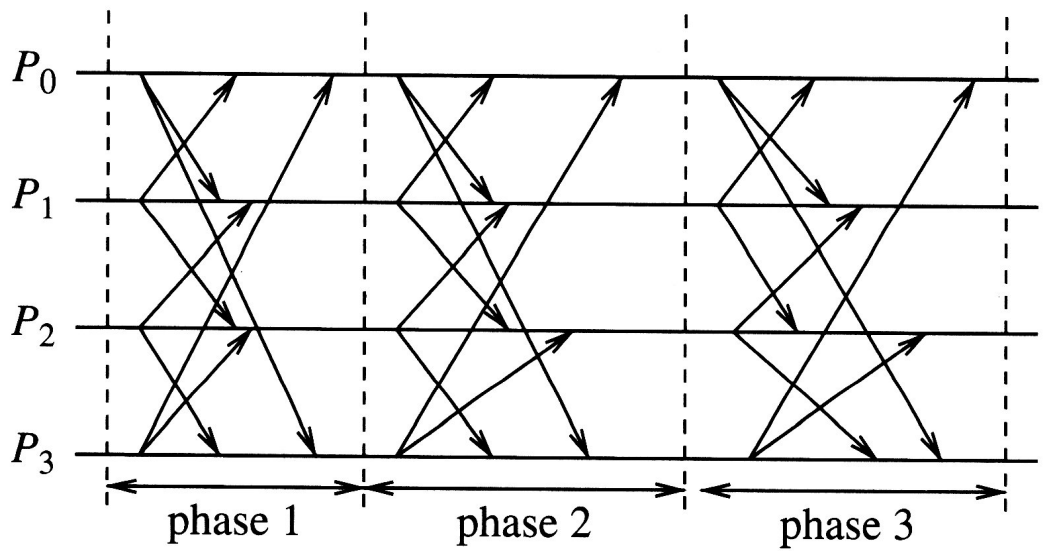
restricted communication time

computation cluster



Parallel computing

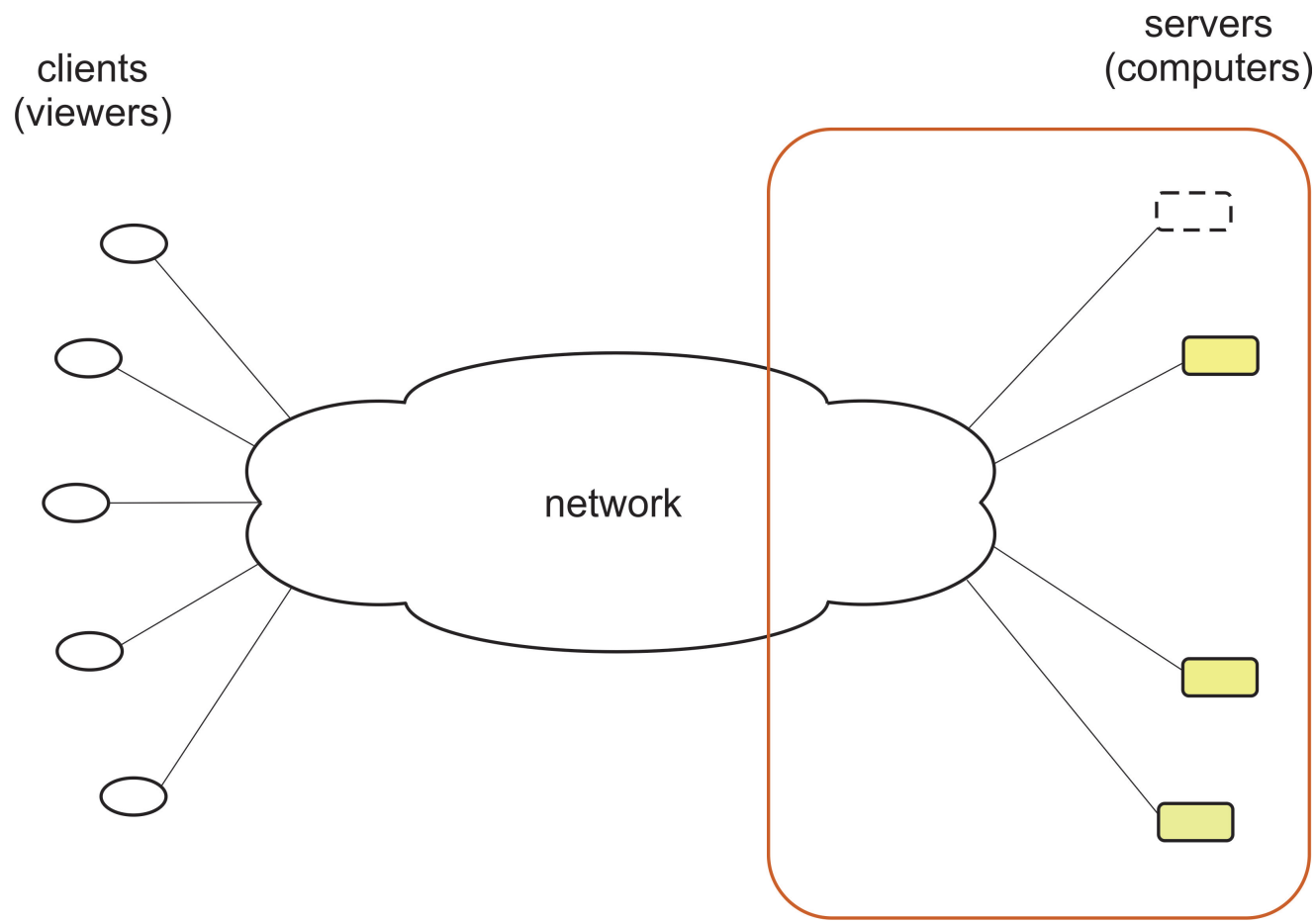
restricted communication time, synchronisation



Distributed computationt

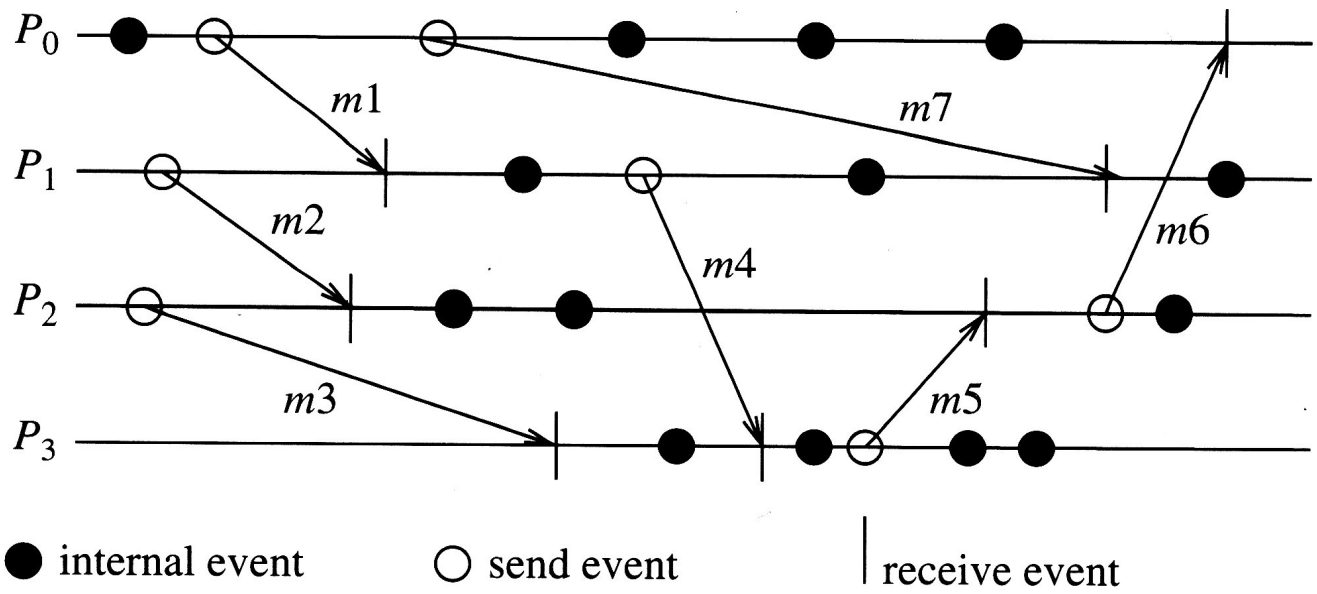
greater communicatioon time

distributed computation



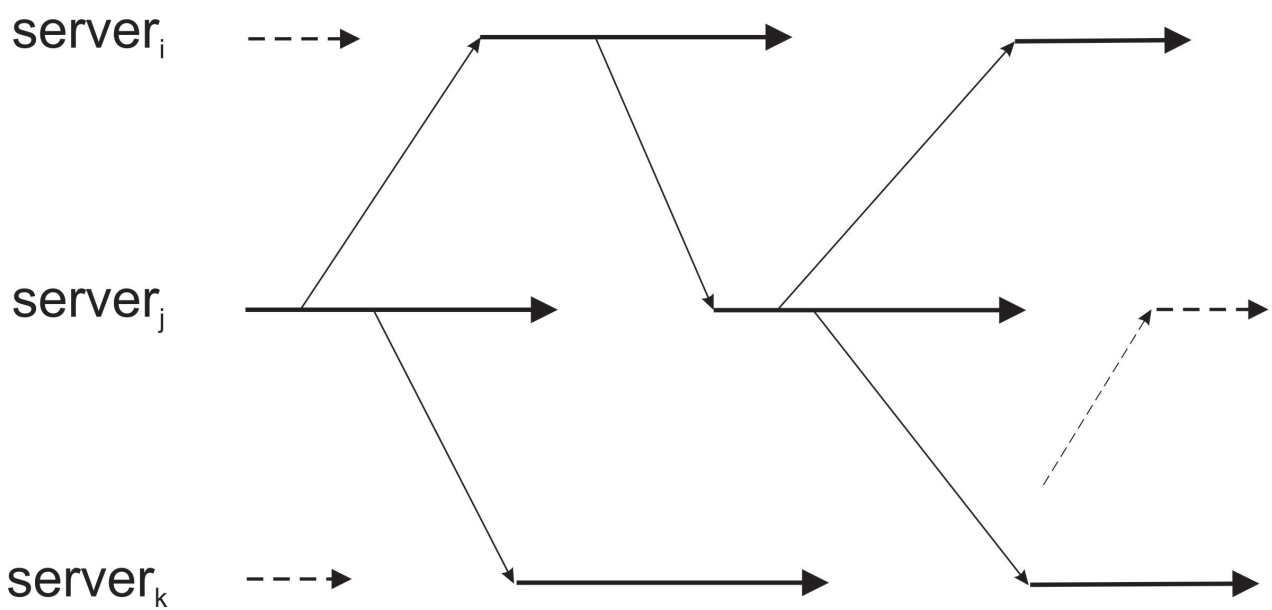
Distributed computation

influence of communication time, independent computation



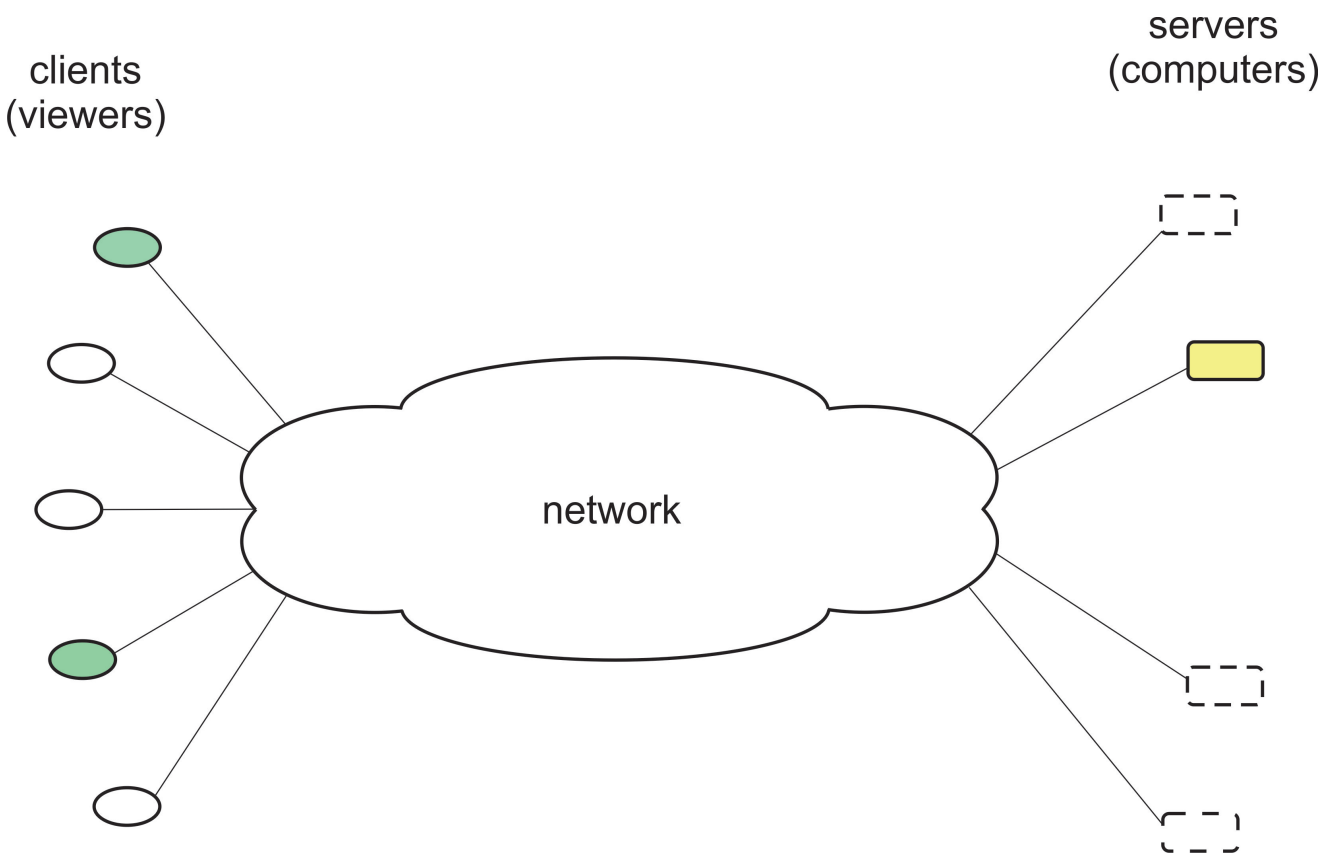
Distributed computation

model



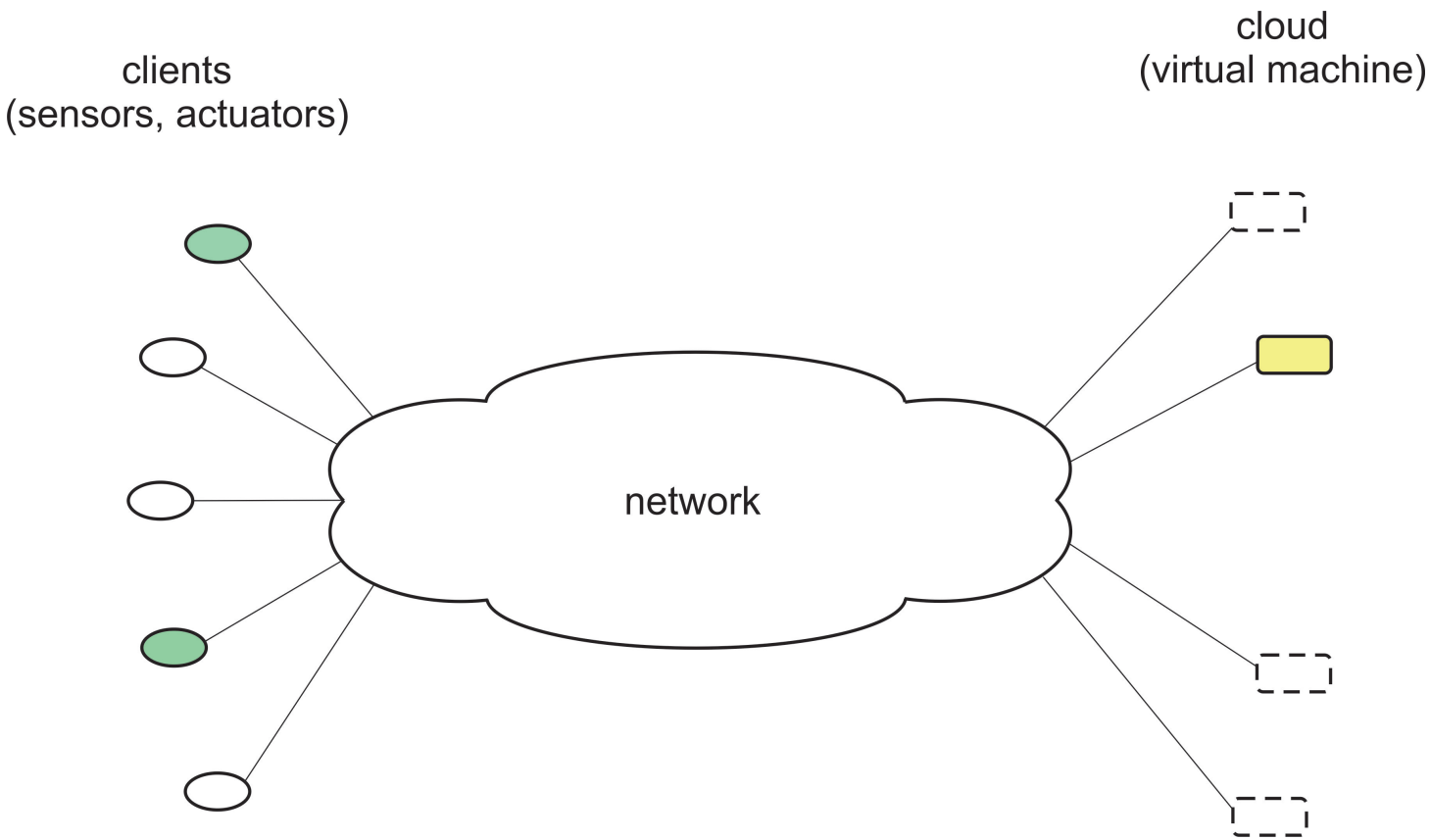
Client-server

"selectivness" of the server, dynamics of relatins



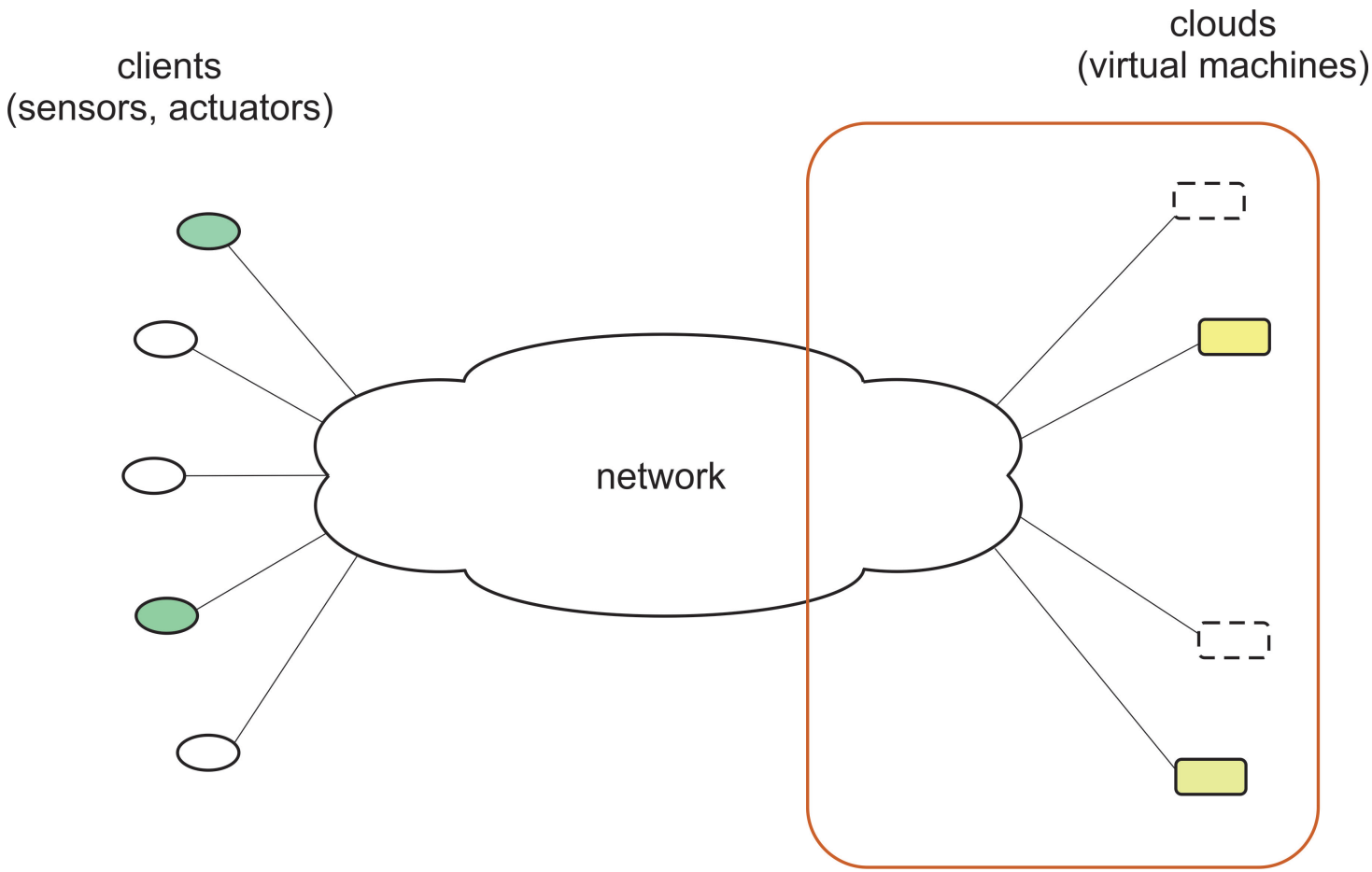
Cloud

selective cloud



Cloud

P2P cloud



Distribution in systems and applications

Motivation

- information exchange
- resource sharing
- increasing reliability through replication
- increased performance through paralelization
- simplification of design through specification

Application structure, communication processes

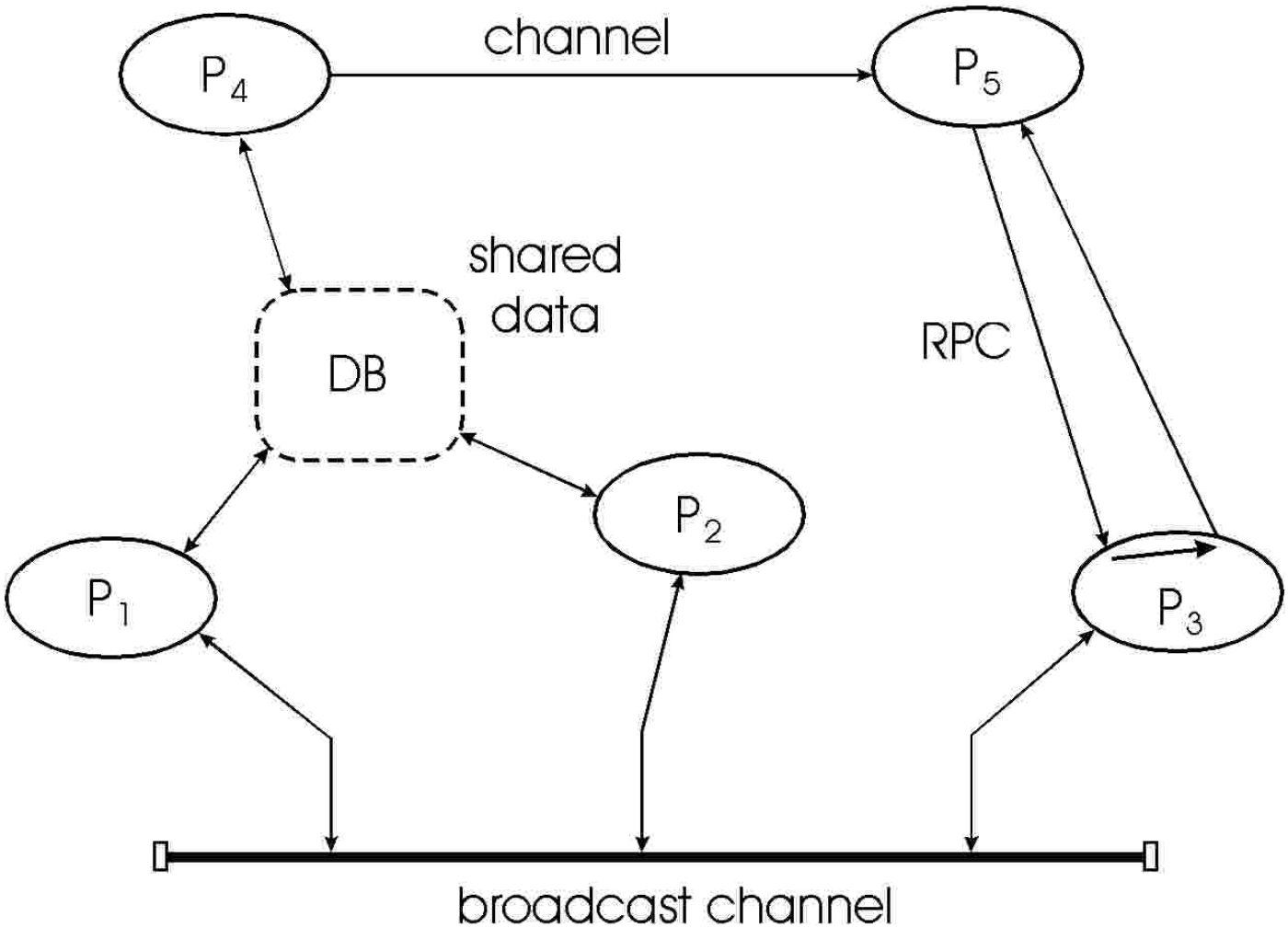
Communication mechanisms for processes

- simple message exchange
- exchange of messages in groups of processes
- messages ammong processes a data areas
- messages in client-server applications

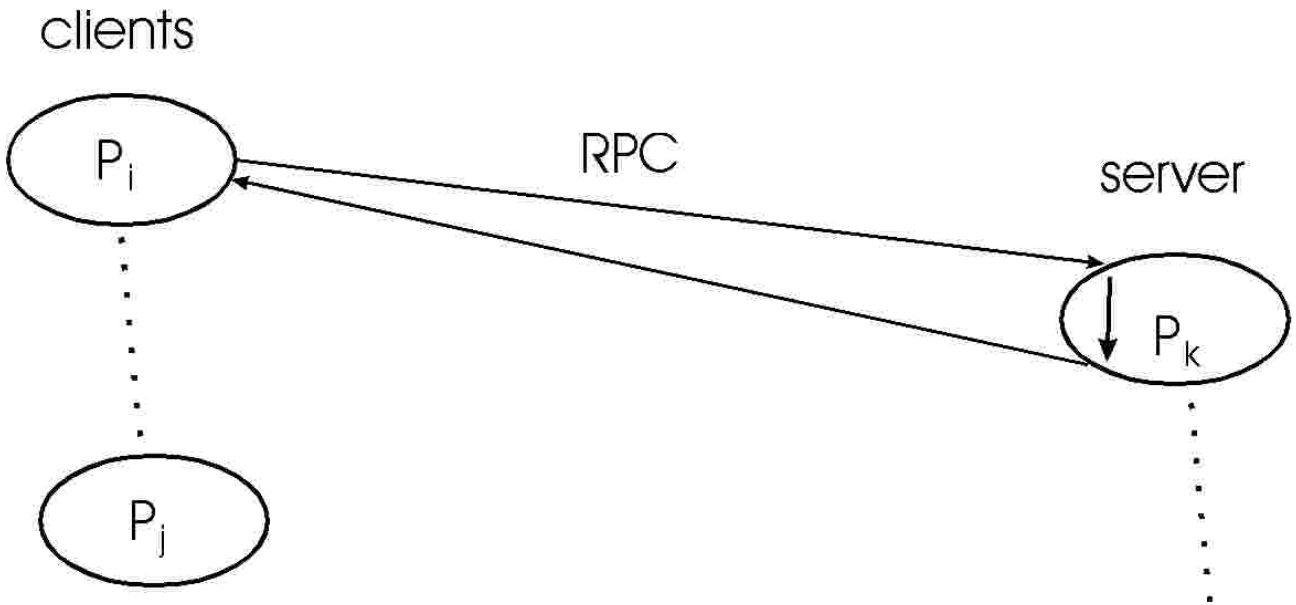
Simple communiccation methods

- asynchronous exchange of messages
- synchronous exchange of messages

Communication mechanisms for processes



Client - server (WEB) services



Message exchange

asynchronous communication

- direct
- indirect

reception on more channels

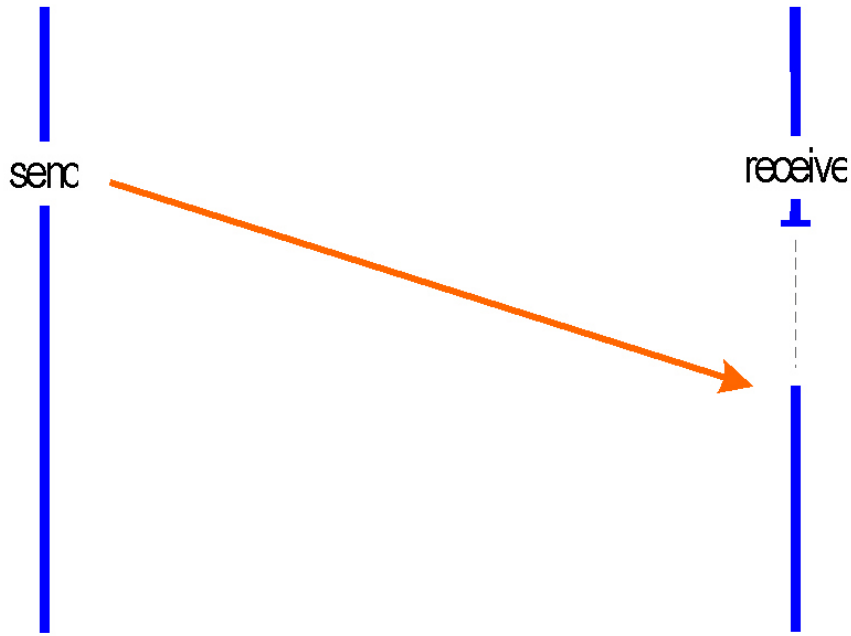
- simple select
- guarded select

synchronous communication

shared channels

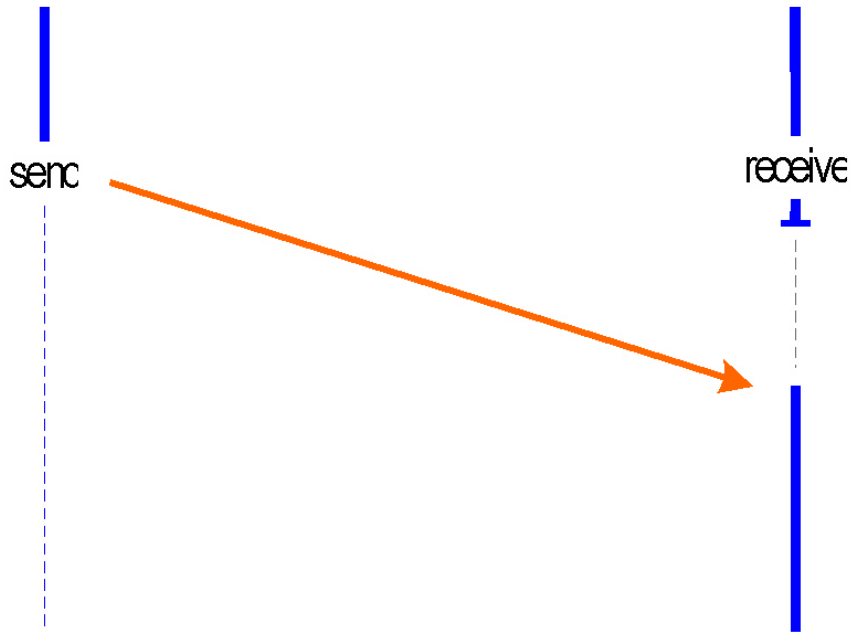
Asynchronous message exchange

Nonblocking sender



Asynchronous message exchange

Blocking sender



Asynchronous message exchange

Direct communication

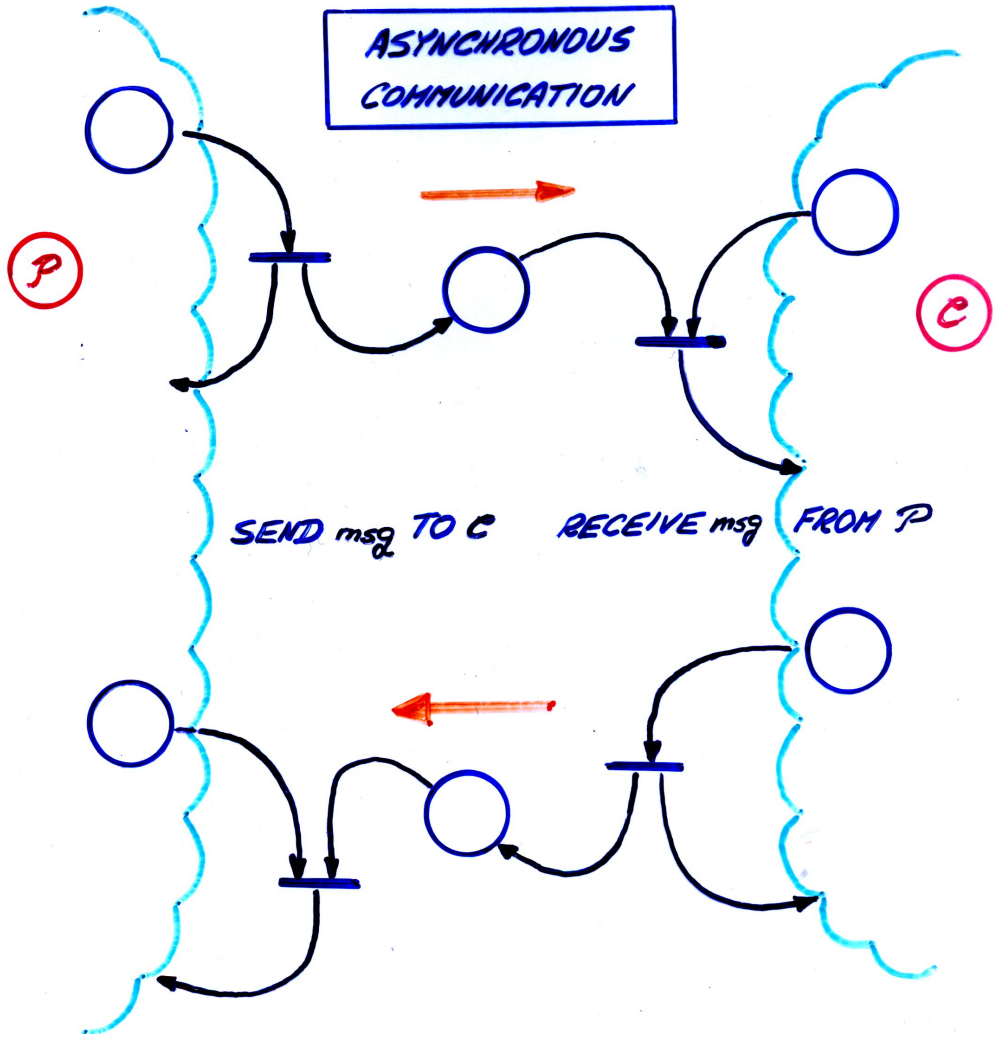
- message sending

  send expr to receiver

- message reception

  receive var from sender

Direct asynchronous message exchange



Asynchronous message exchange

Indirect communication

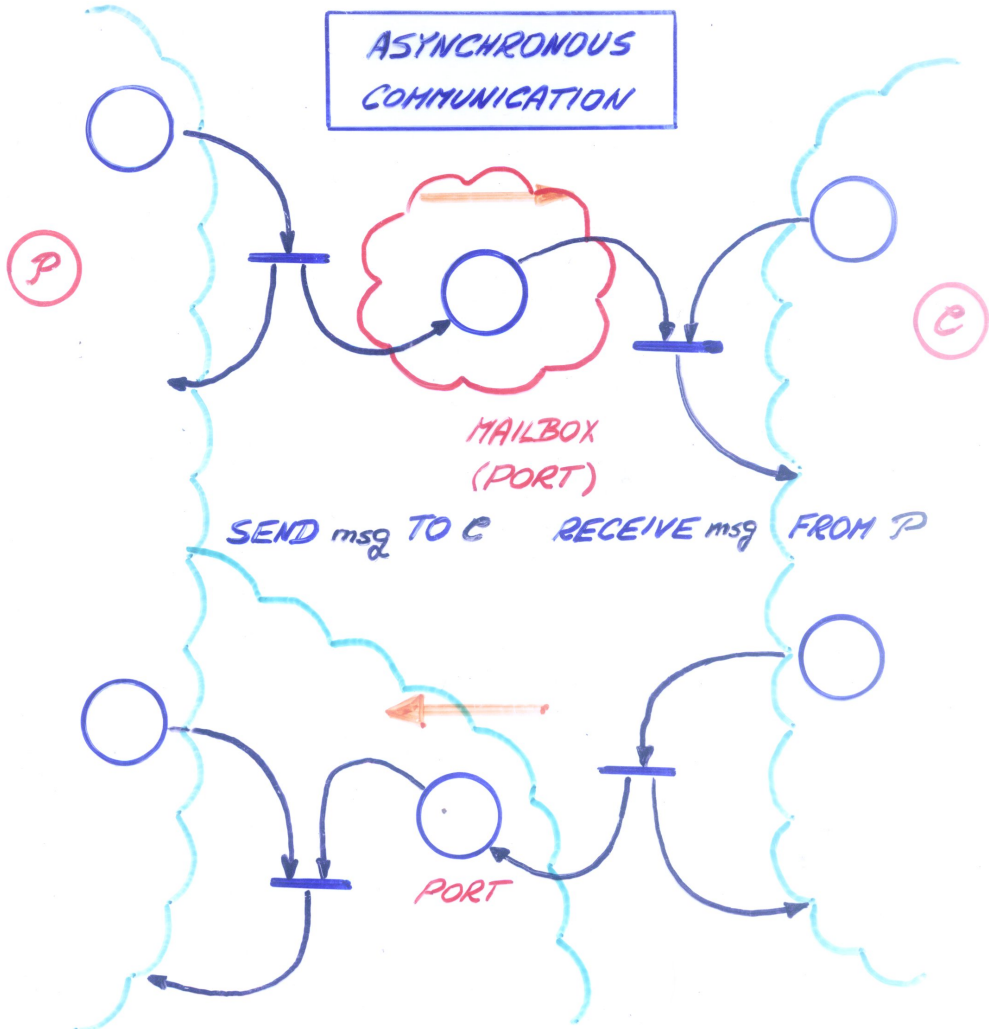
- message sending

  send expr to channel

- message reception

  receive var from channel

Indirect asynchronous message exchange



Reception on more channels

Simple select

select
receive msg_1 from ch_1 => $stat_1$
or receive msg_2 from ch_2 => $stat_2$
or receive msg_3 from ch_3 => $stat_3$
end

Reception on more channels

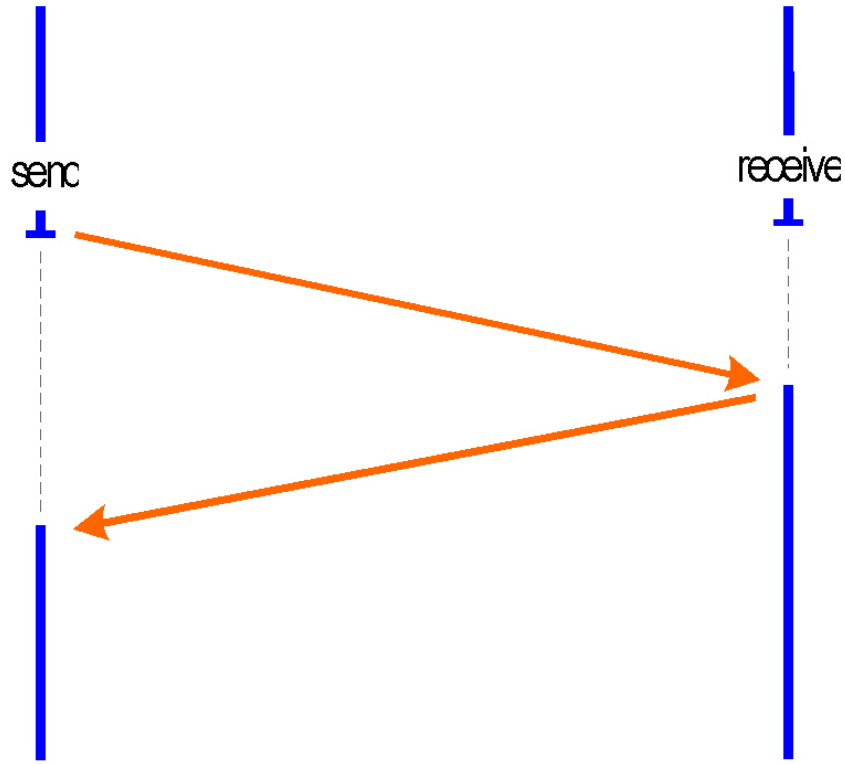
Guarded select

select

when $comd_1$ receive msg_1 from ch_1 => $stat_1$
or when $comd_2$ receive msg_2 from ch_2 => $stat_2$
or when $comd_3$ receive msg_3 from ch_3 => $stat_3$

end

Synchronous message exchange



Synchronous message exchange

Direct (synchronous) communication

- message sending

  send E to R

- message reception

  receive V from S

Synchronous message exchange

Hoare CSP - Communicating Sequential Processes

Direct (synchronous) communication

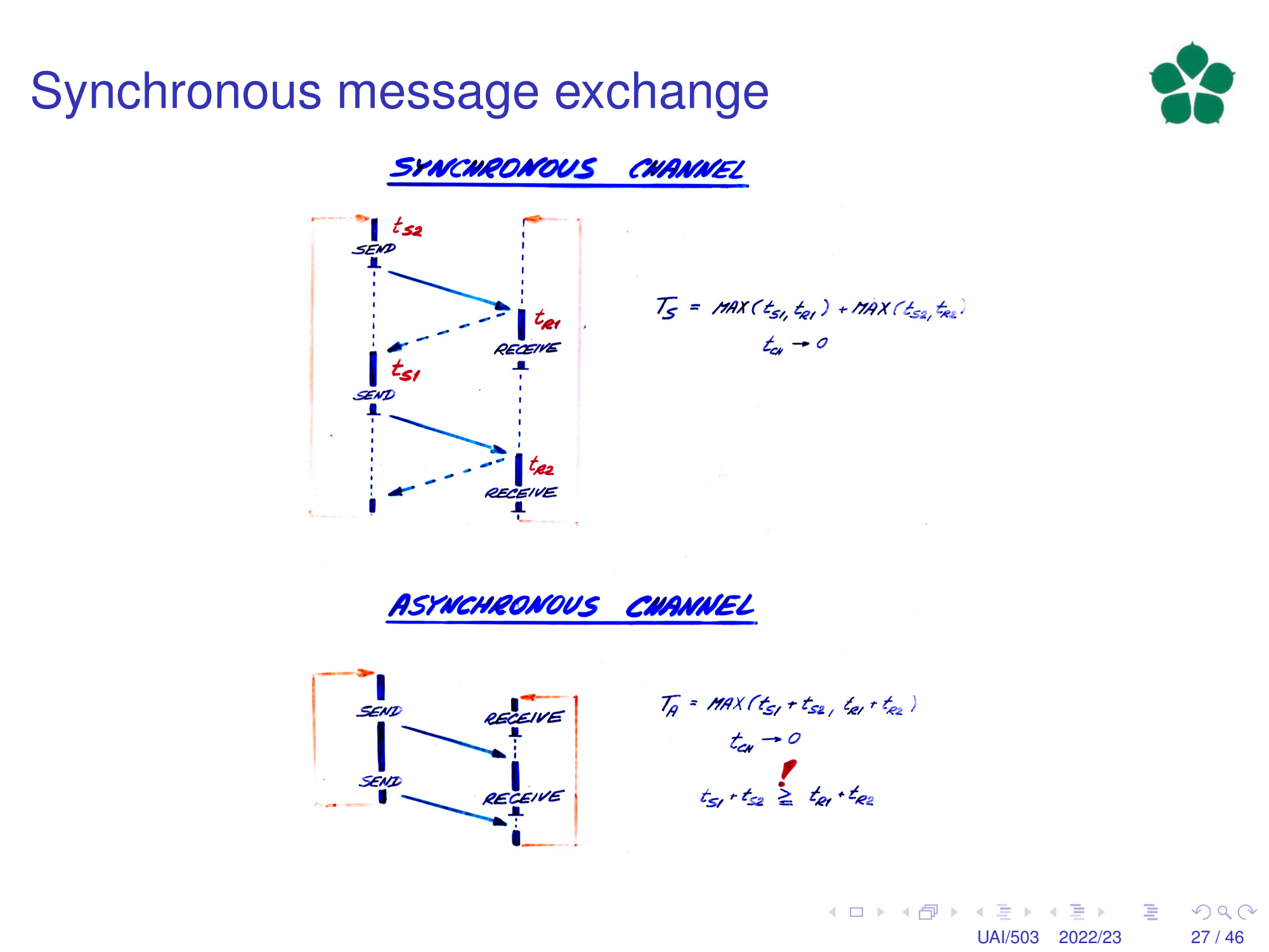
- message sending

  E ! R

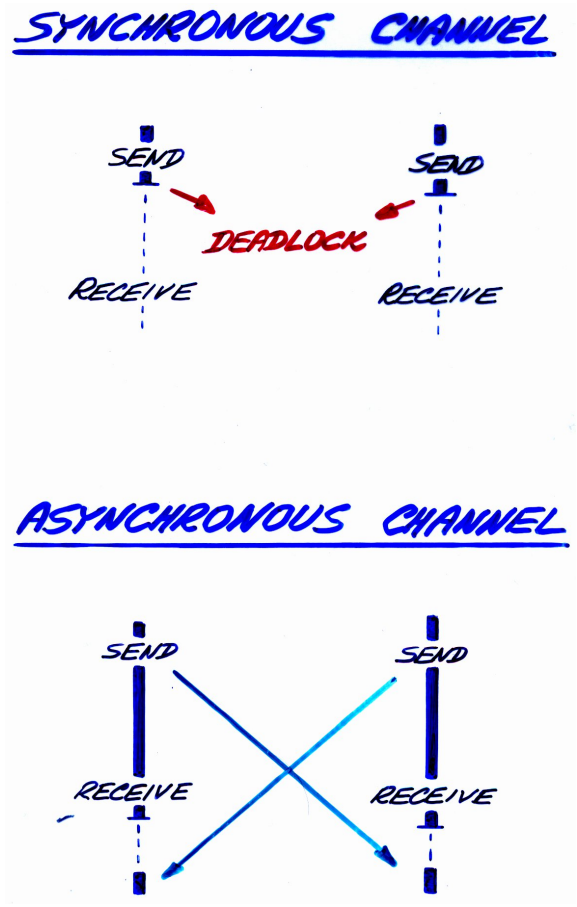
- message reception

  S ? V

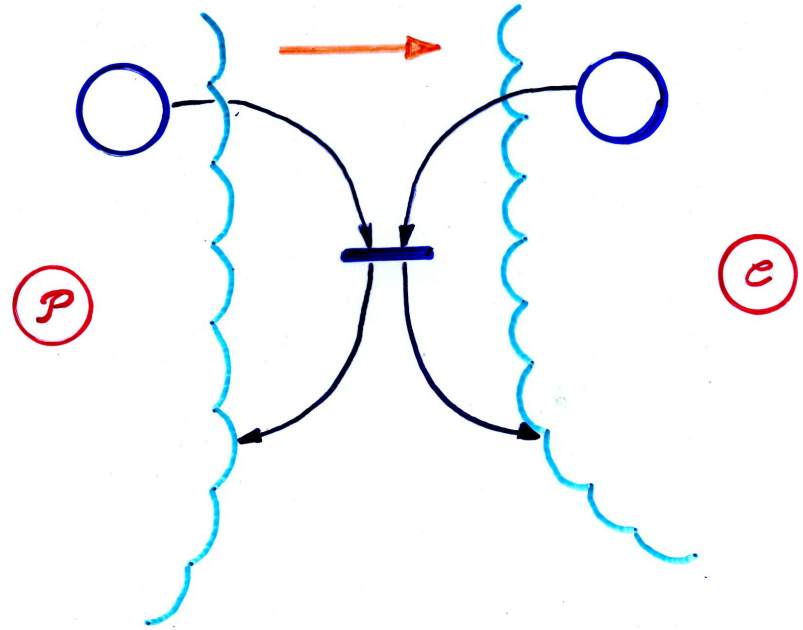
Synchronous message exchange



Synchronous message exchange



Synchronous message exchange



CSP

C ! expr → P ? var

```
P::
[
  *[
    ] B!C
]
```

```
B::
[ POOL : (1..100) CHAR;
  I,O : 1..100; C : 0..100;
  I := 1; O := 1; C := 0;
  *[ C < 100; P?POOL(I) → I := (I MOD 100) + 1
   [] C > 0; C?MORE → C!POOL(O);
                      O := (O MOD 100) + 1
   ]
]
```

```
C::
[
  *[ B!MORE
   ] B?C
]
```

HTML a XML technologies

HyperText Markup Language, eXtensible Markup Language

HTML technologie

- WWW pages
- CGI skripts
- Java servlets
- JSP skriptlets

XML technologies

- XML dokuments
- DOM / SAX parsing
- XSLT a XPath tools

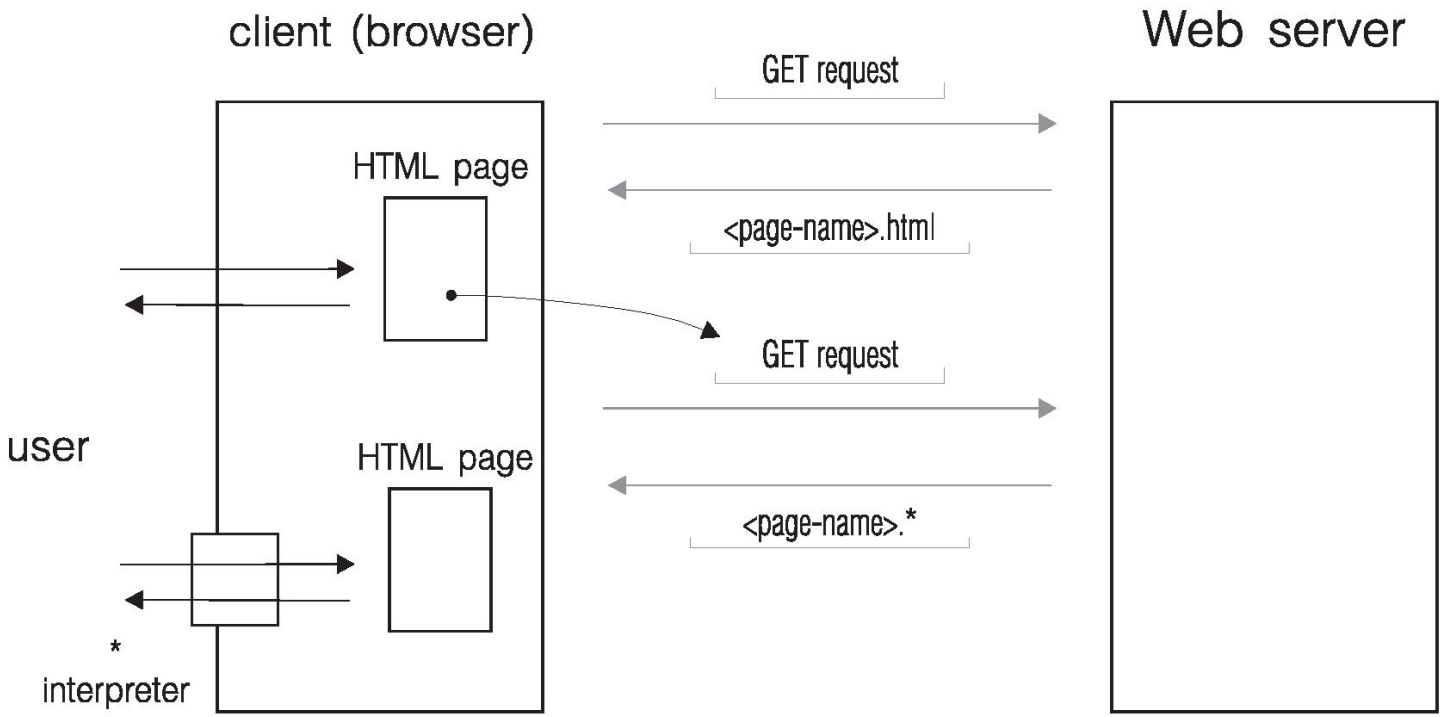
HTML

HyperText Markup Language

```
<html>
    <head>
        <title> HelloWorld </title>
    </head>

    <body>
        <h1> Header<h1>

        <p>
            Text . . .

        <table>
            <tr>
                    . . .
            </tr>
        </table>

        <a href="page-a/index.html">Page A </a>
    </body>
</html>
```

HTML page

HyperText Markup Language



HTML page

HTML request

http: \\ dsn.felk.cvut.cz\HelloWorld.html

HTML response

```
<!DOCTYPE html PUBLIC "- //W3C//DTD HTML 1.0 ...
<html>
    <head>
        <title> HelloWorld </title>
    </head>

    <body>
        <h1> Hello, world! </h1>
    </body>
</html>
```

HTTP protocol

HyperText Transfer Protocol

HTTP request

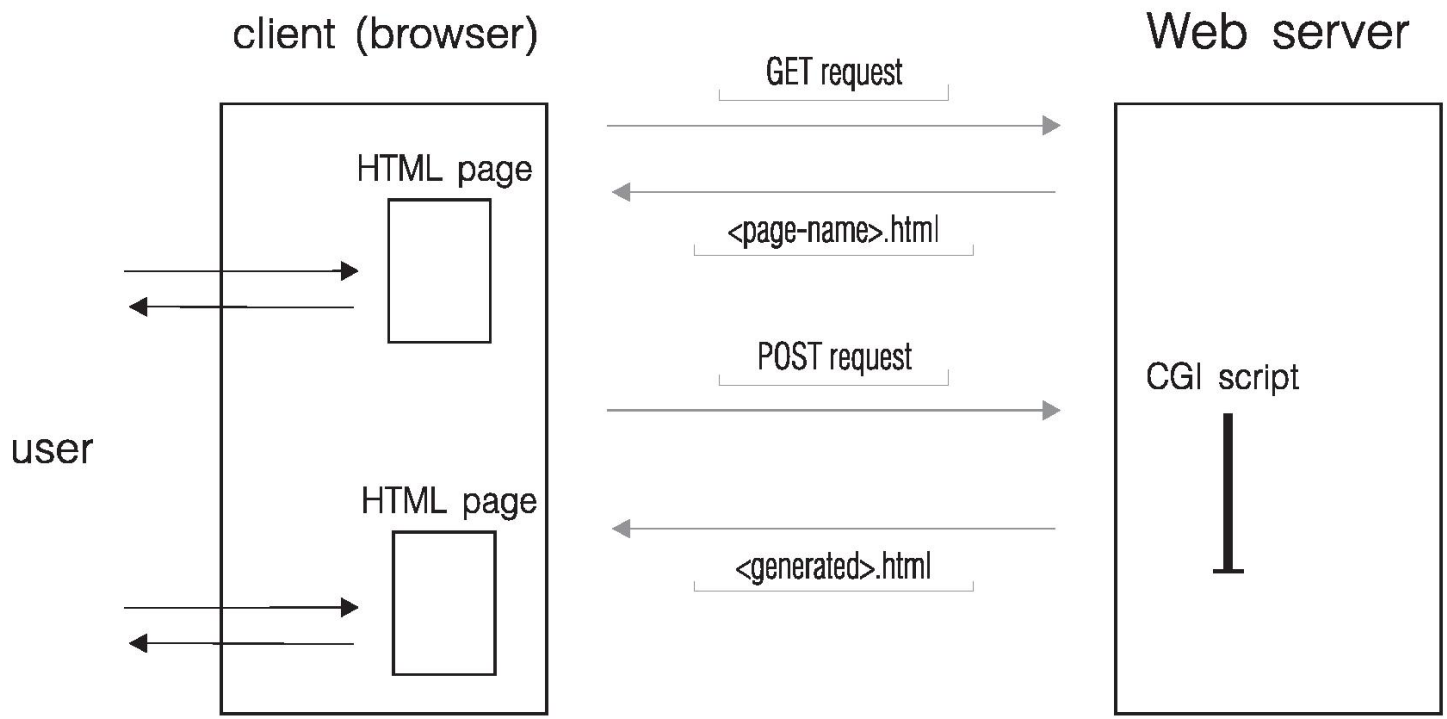
GET /index.html HTTP/1.1
Host: dsn.felk.cvut.cz

HTTP response

HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8

CGI script

Common Gateway Interface



CGI script

Common Gateway Interface

Method GET

http://dsn.felk.cvut.cz/HelloWorld.cgi

```
#include <stdio.h>
...
int main(void)
{
  printf("Content-Type: text/plain \n \n");
  printf("Hello World in C! \n");
}
```

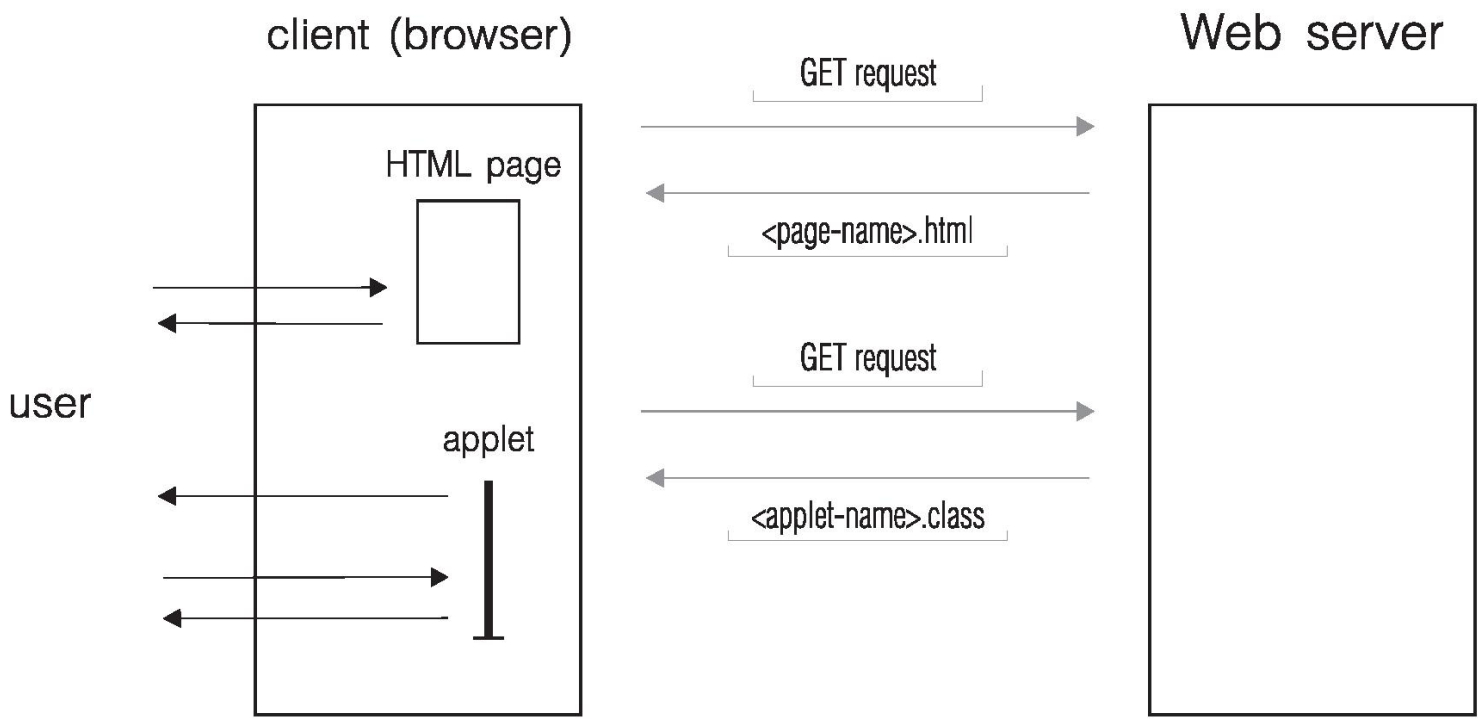
CGI script

Common Gateway Interface

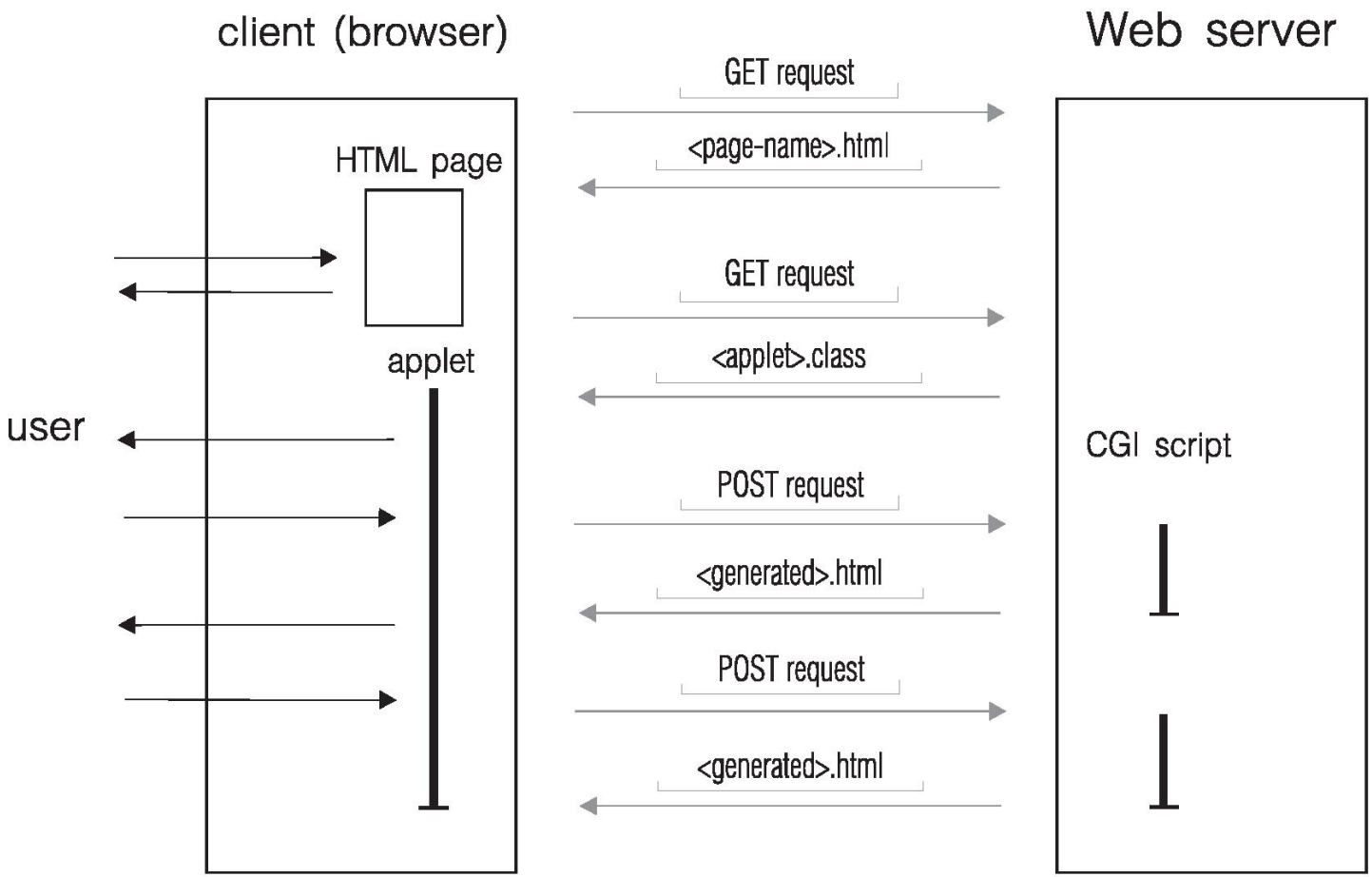
Method POST

```
POST /addMessage HTTP/1.0
Host: www.mailtothefuture.com
Content-type: application/x-www-form-u
Content-length: 133

        data
```

Applet


client (browser)
Web server
GET request
HTML page
<page-name>.html
user
GET request
applet
<applet-name>.class

Applet + CGI script



XML - eXtensible Markup Language

```
<?xml version="1.0"encoding="UTF-8">
    <dokument>
        <data>
                some data . . .
        </data>
        . . . only comment . . .
    </dokument>
```

XML - example

```
<qa:book xmlns:qa="http://www.qa.com">

    <qa:title>A Few Good Men</qa:title>

    <qa:lentTo maxLoan="28">
        <B>Doe</B>, John
    </qa:lentTo>

</qa:book>
```

XML - document's structure definition

DTD - Data Type Definition

```
<!DOCTYPE booklist [

<!ELEMENT booklist (book)+>
<!ELEMENT book (person, ...)>
<!ATTLIST book maxLoan CDATA #REQUIRED>
<!ELEMENT person (#PCDATA)> ]>
```

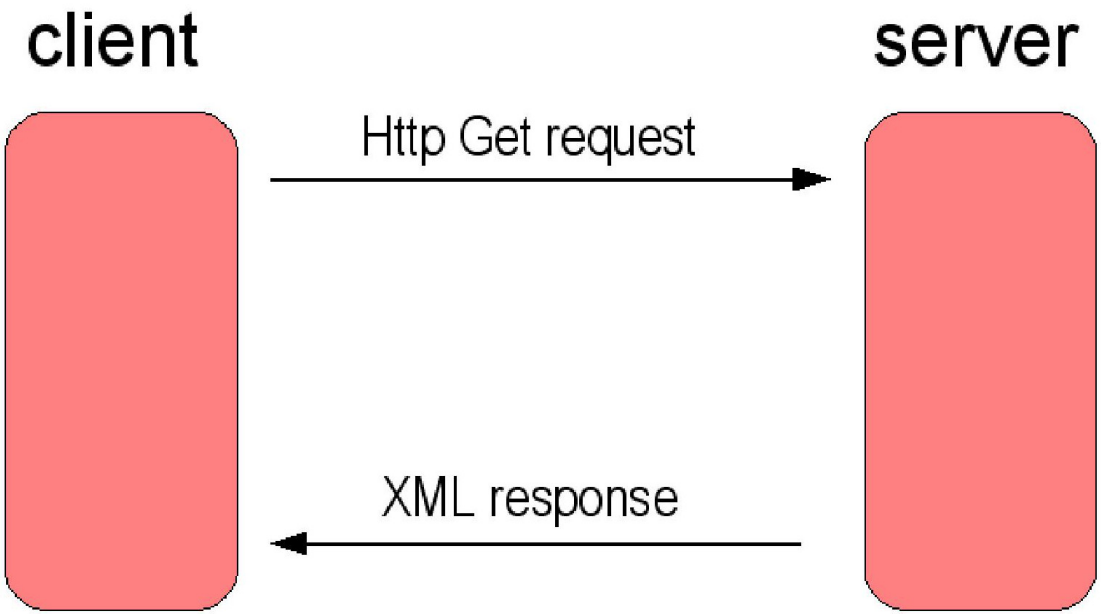
XML Schema

- replaces currently archaic DTD
- XML Schema serves for type definition

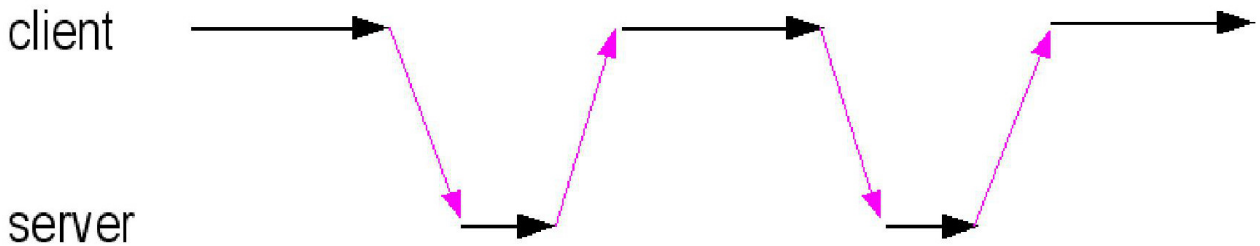
XML Schema - type definition

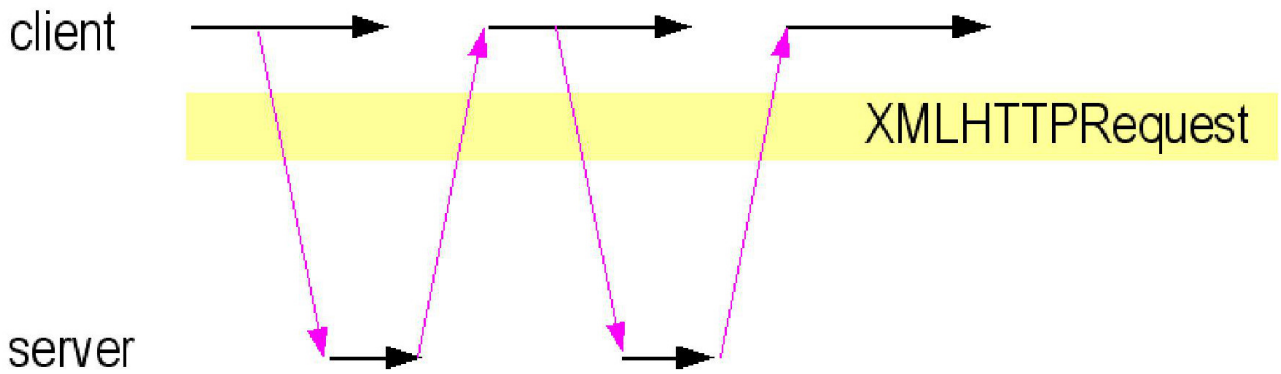
```
<?xml version="1.0"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="booklist">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="book"    maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="person"    type="xsd:string"/>
              . . .
            </xsd:sequence>
            . . .
          <xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

AJAX - Asynchronous JavaScript & XML



AJAX - Asynchronous JavaScript & XML





Node.js - Node JavaScript