# A2 Verteilte Systemsoftware
# Part 1 – Intro "Distributed Systems"

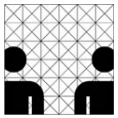
aus: Coulouris / Dollimore / Kindberg: "Distributed Systems: Concepts and Design", Addison Wesley/Pearson Internat. , 5. Ausg., 2012

***"Distributed systems"* - simple definition: HW+SW components, located on global networks, which communicate & coordinate their actions only by message passing; important: *common global goal*!**

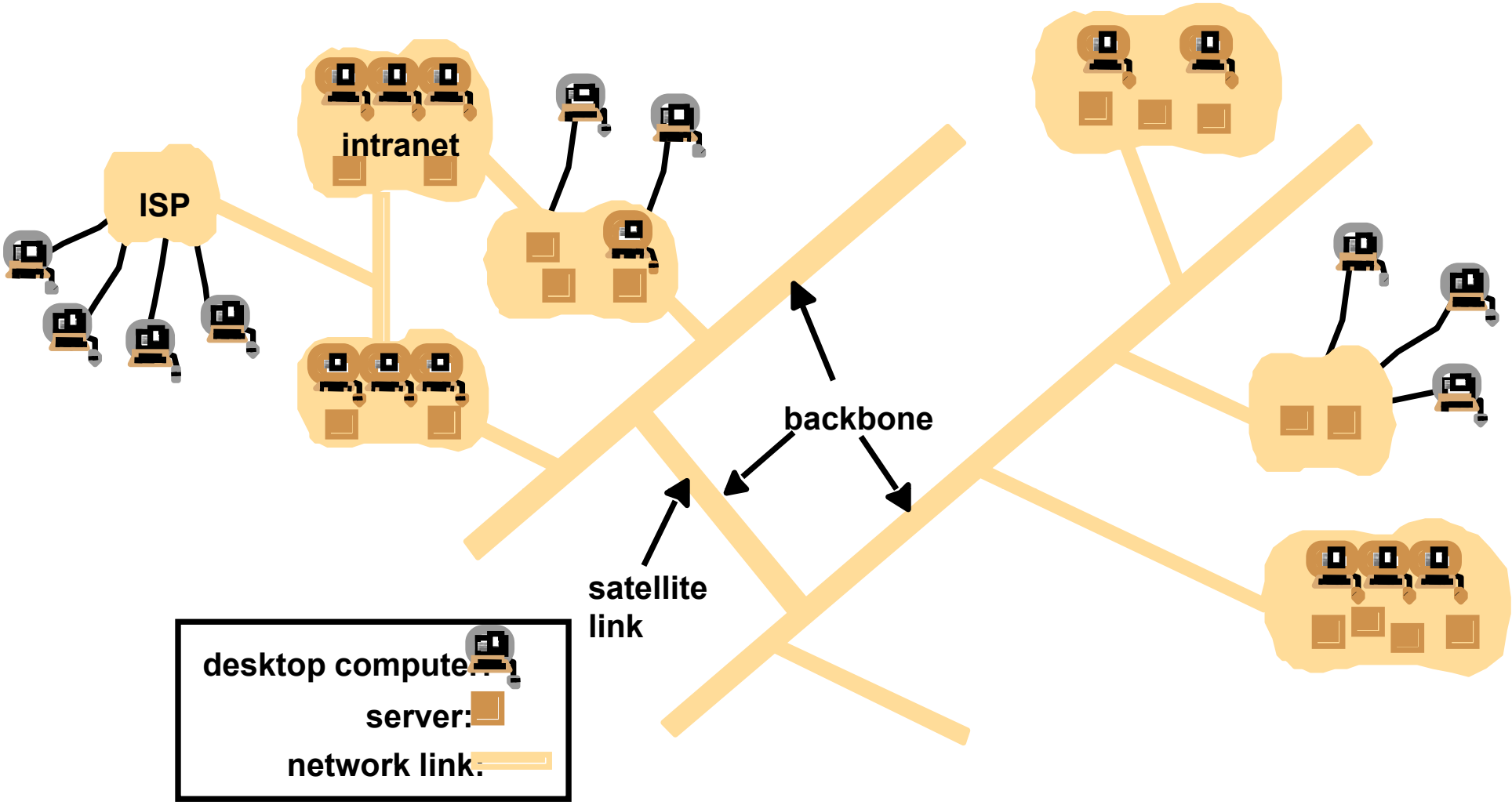
**Typical characteristics:**

- ***Concurrency:*** → **siehe "Nebenläufigkeit & Verteilung"**
- ***No global clock:*** → **s. "Synchronisation" (+ Vorl. "VIS" später)**
- ***Independent failures:*** → **alle Kapitel + "Transparenz"-Eigen-schaften**

**Beispiele:**

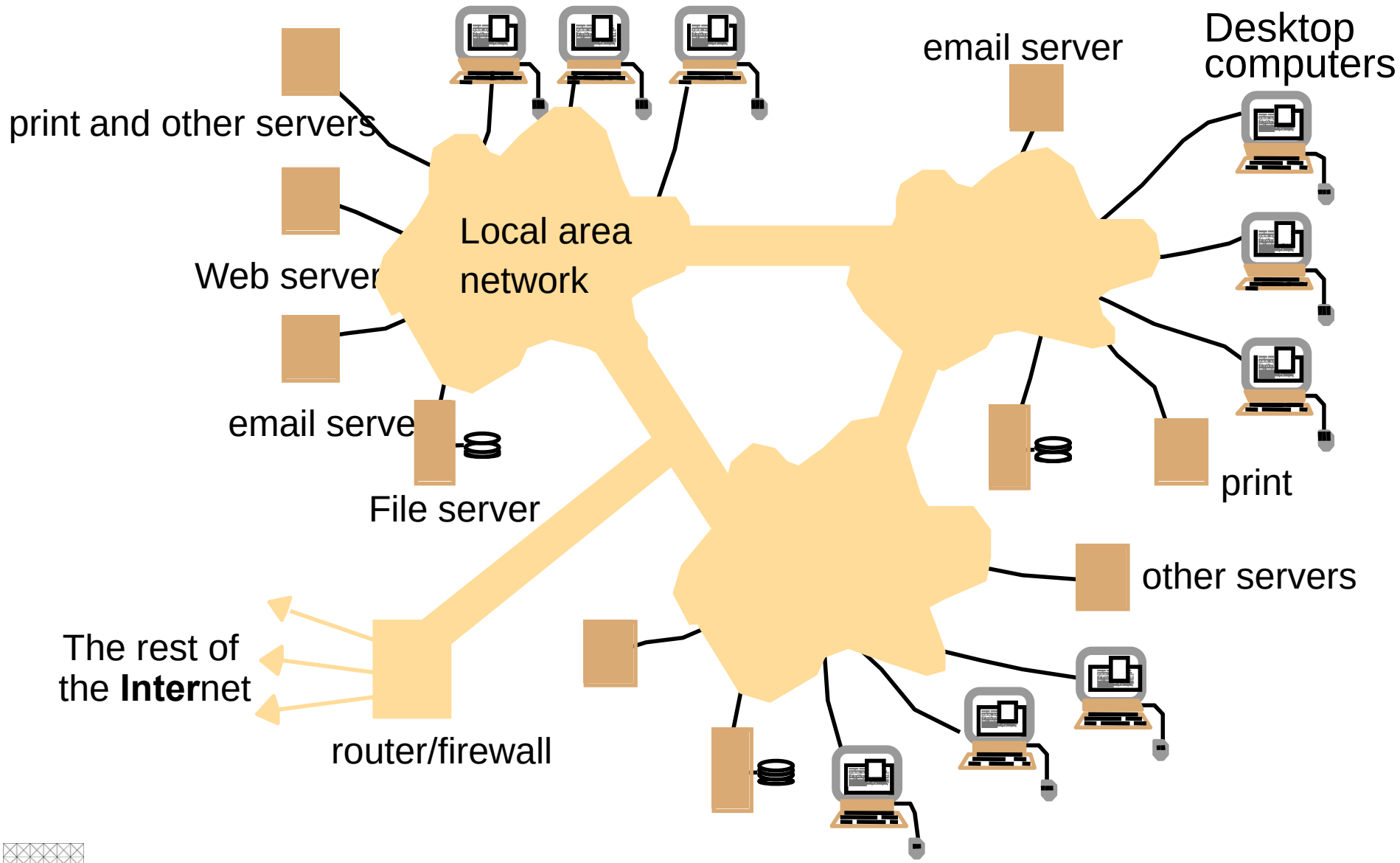


# Distributed Systems: *"A typical portion of the **Inter**net"*

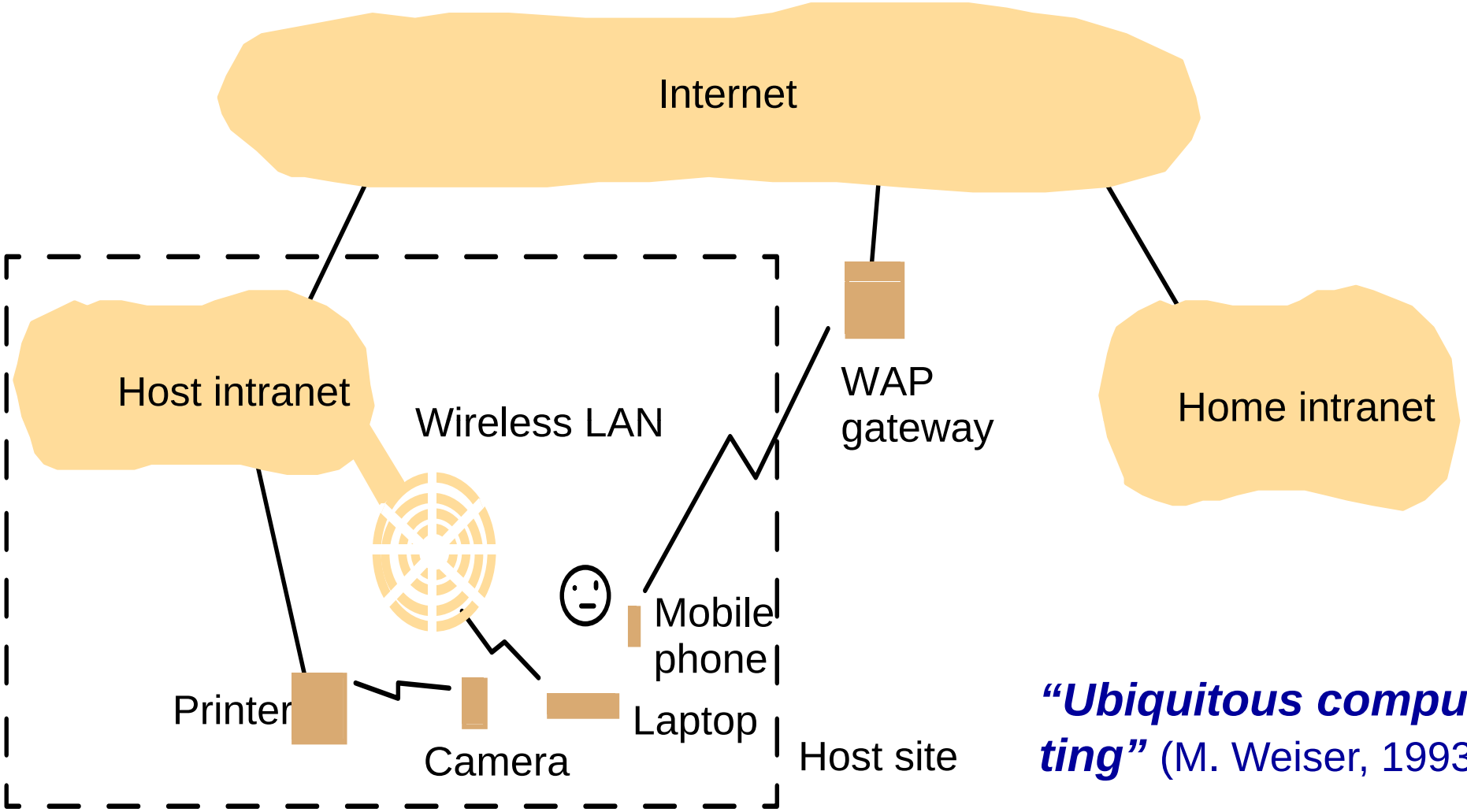



**Communication, System software (i.e. Operating System), Middleware, Security**

# A Typical Part of an *Intra*net

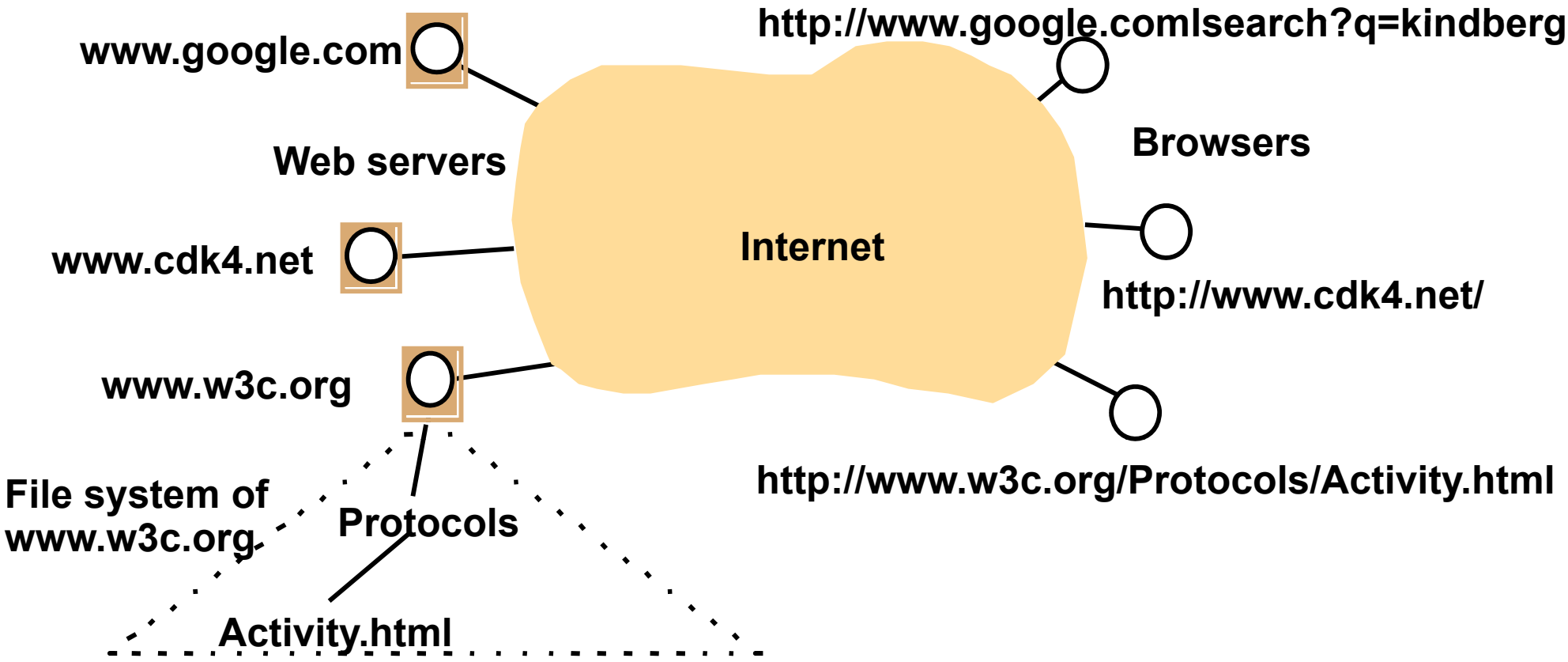


# “Mobile Systems”:
## Portable and handheld devices in a distributed system

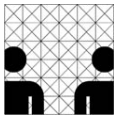



***“Ubiquitous computing”*** (M. Weiser, 1993)

# (Web) Server: Web servers and web browsers




**Ressource sharing in the web:** ***Servers, clients, remote service invocation, resource location (URL), interfaces (HTML), protocols (http), web services, service-oriented computing/ architecture (SOA)***



# Number of Computers in the Internet

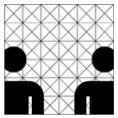
## Challenges:

- *Heterogeneity*
- *Openness*
- *Scalability*
- *Security*

| *Datum* | *# Computer (mit reg. IP-Addr.)* |
|---|---|
| 1979, Dez. | 188 |
| 1989, Juli | 130,000 |
| 1999, Juli | 56,218,000 |
| 2003, Jan. | 171,638,297 |
| …. | …….. |
| 2011 | über 2 Mrd. IN Benutzer |
| …. | |

**Inzwischen:**

**“Internet of Things”**



# Distributed Systems Goals: *"Transparencies" (1)*

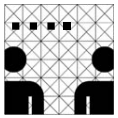
***Overall goal: "Distribution Transparency" – can be subdivided into:***

***Access transparency:*** **enables *local and remote* resources to be accessed using *identical* operations.**

***Location transparency:*** **enables resources to be accessed *without knowledge of their location*.**

***Concurrency transparency:*** **enables several processes to operate *concurrently* using shared resources *without interference* between them.**

***Replication transparency:*** **enables multiple instances of resources to be used to increase reliability and performance *without knowledge of the replicas* by users or application programmers.**



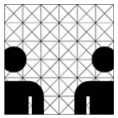
# Distributed Systems Goals: *"Transparencies" (2)*

***Failure transparency:*** **enables the *concealment of faults*, allowing users and application programs to complete their tasks despite the failure of hardware or software components.**

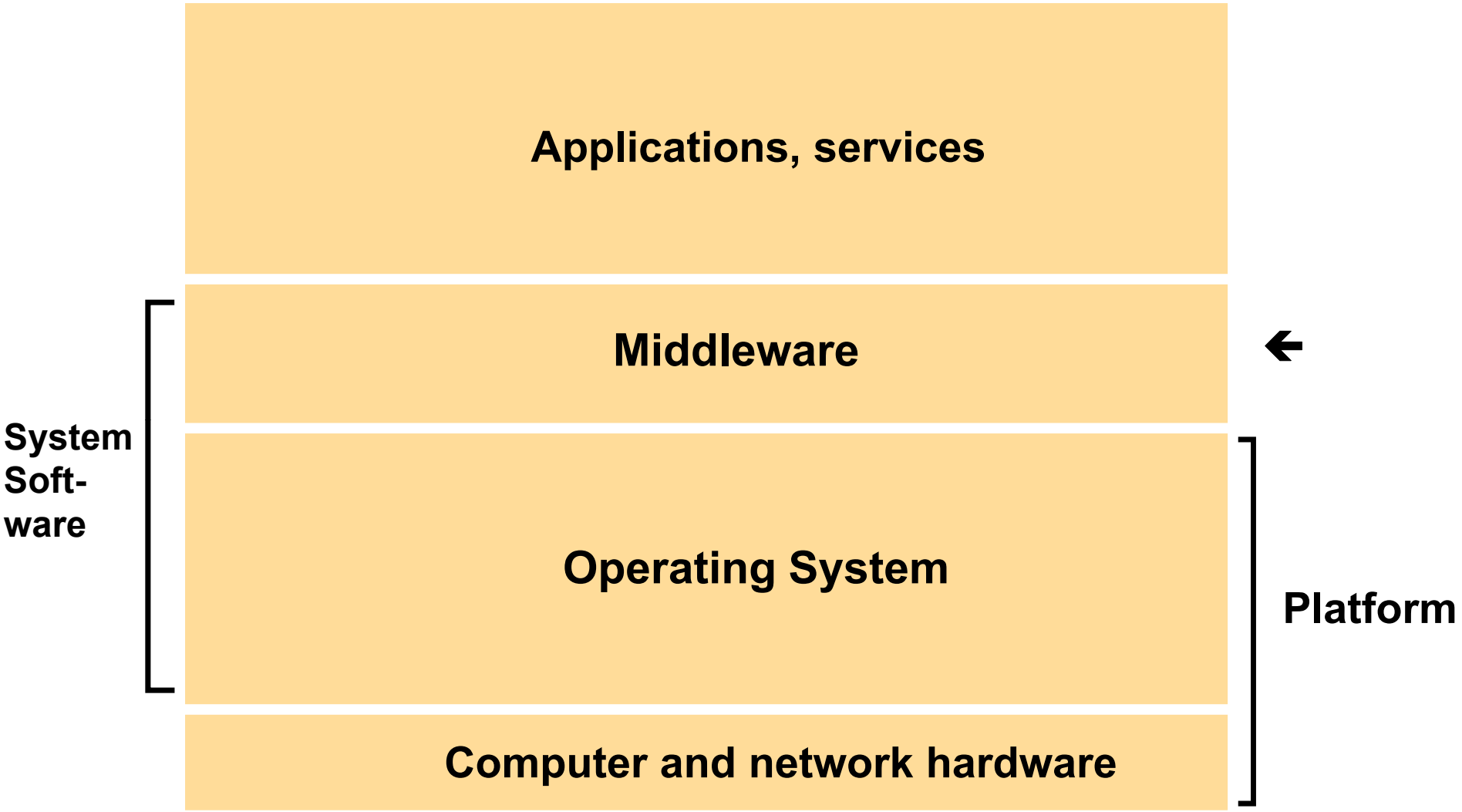
***Mobility transparency:*** **allows the *movement of resources* and clients within a system without affecting the operation of users or programs.**

***Performance transparency:*** **allows the system to be *reconfigured* to improve performance as loads vary.**

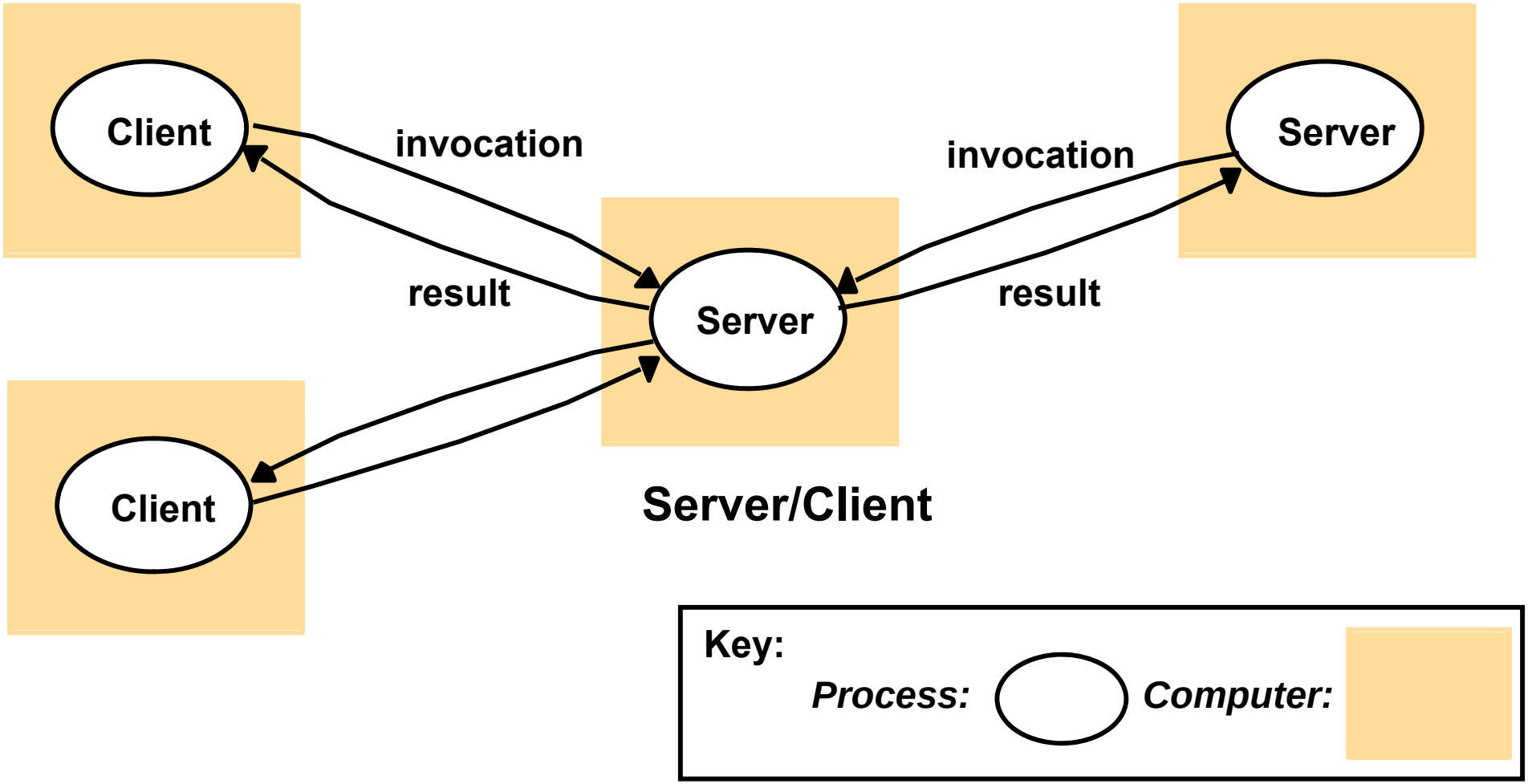
***Scaling transparency:*** **allows the system and applications to *expand* in scale without change to the system structure or the application algorithms.**



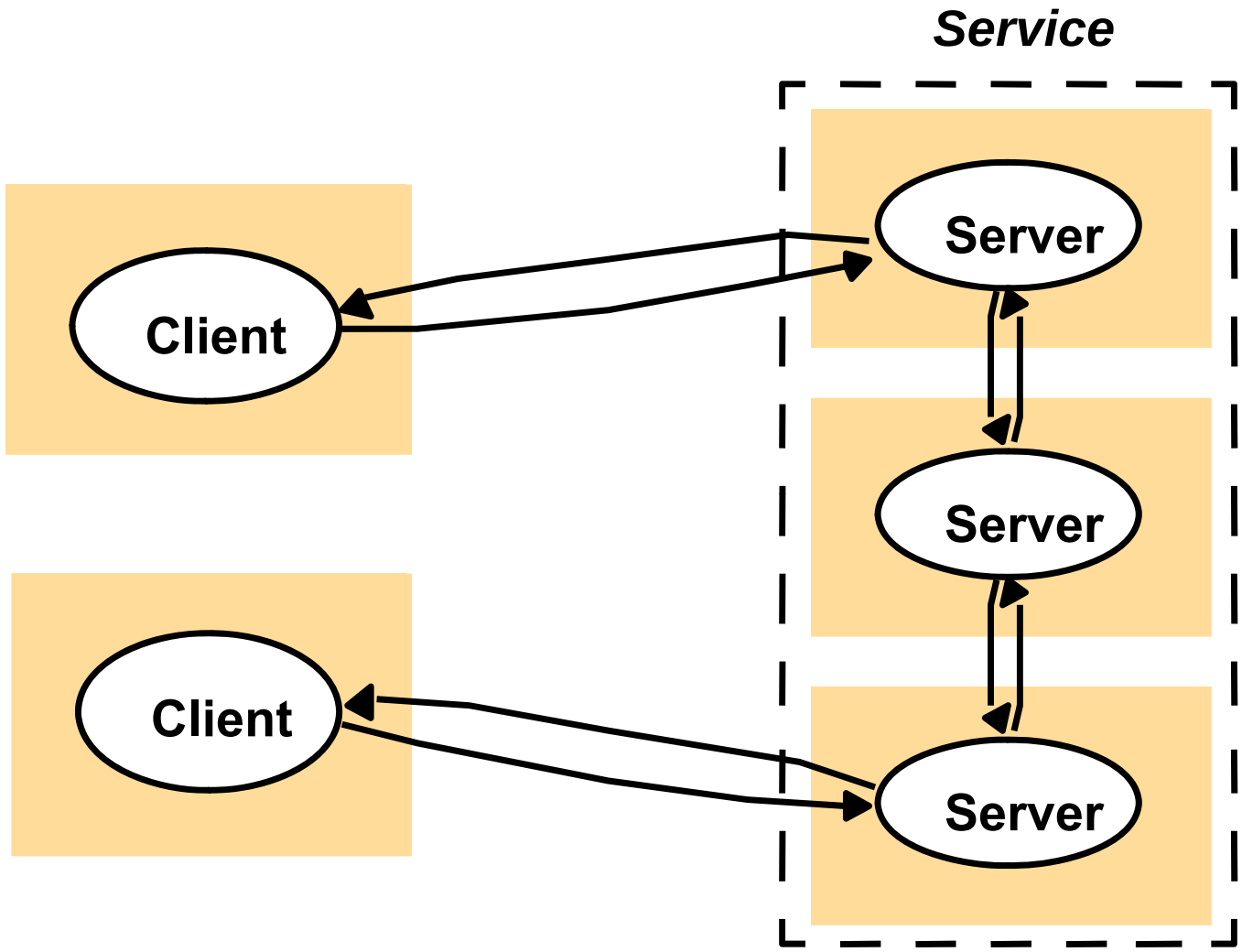
# Part 2 - System Models:
## Software and hardware service layers in distributed systems

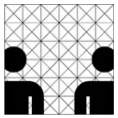


# **Client/Server:** (a) Clients invoke individual servers

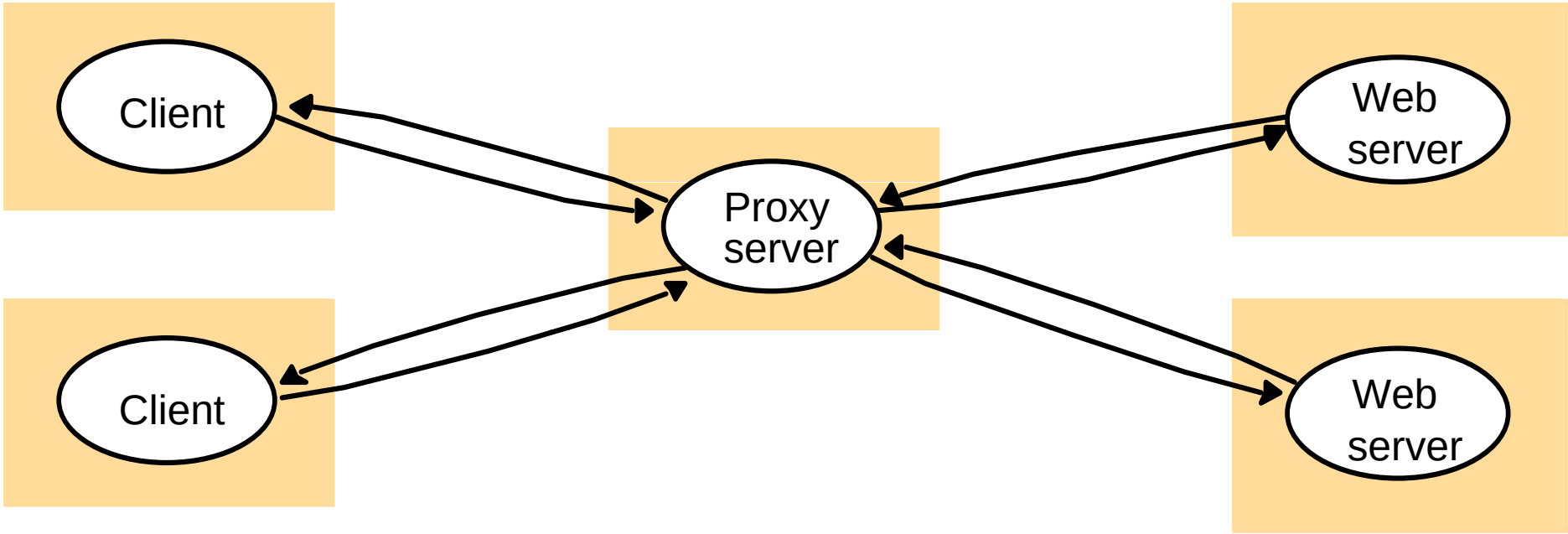


# Client/Server:
## (b) A ***service*** provided by multiple servers



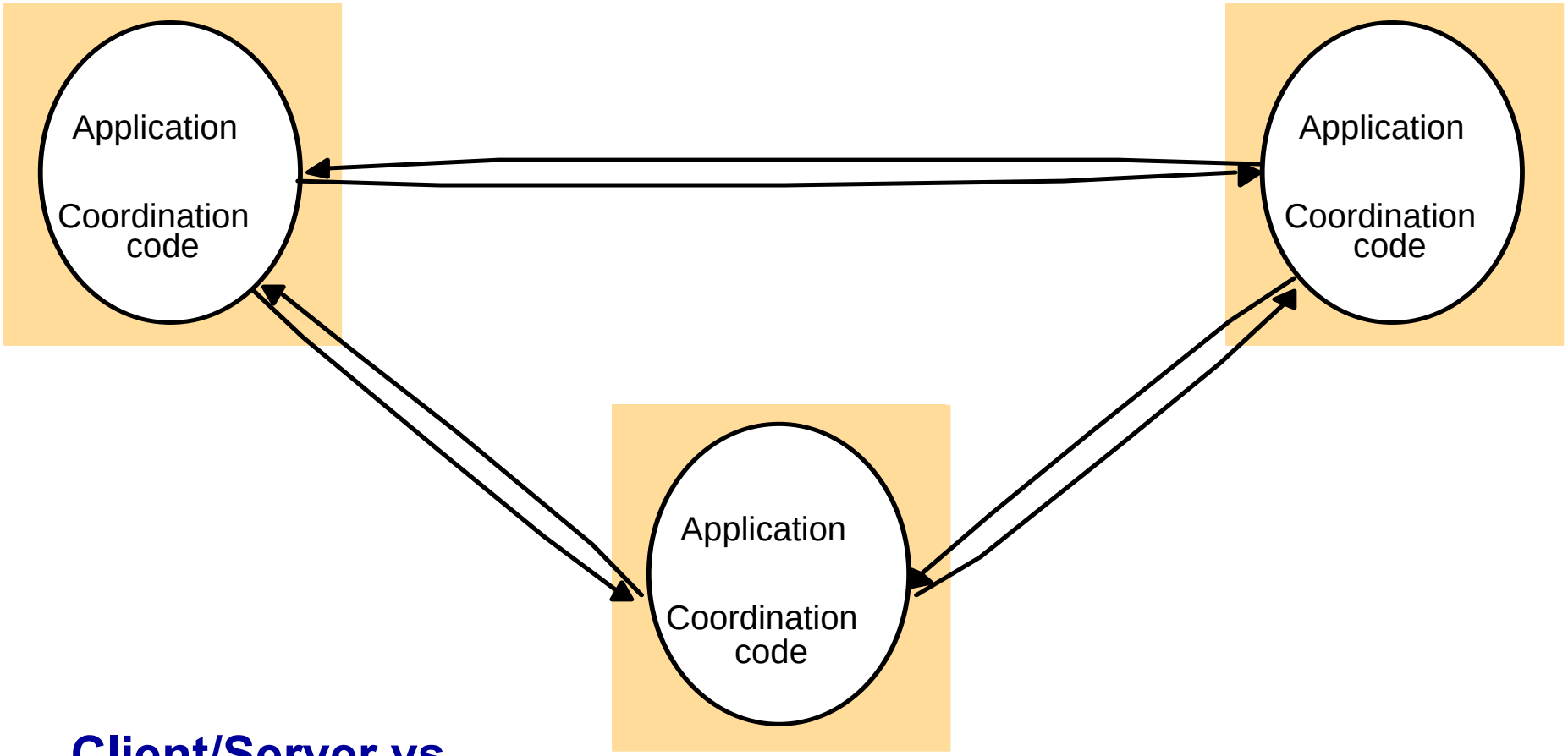



# **Client/Server:** (c) Web proxy server

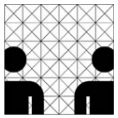



**Proxy server: shared cache of web resources**

# P2P: A distributed application based on peer processes

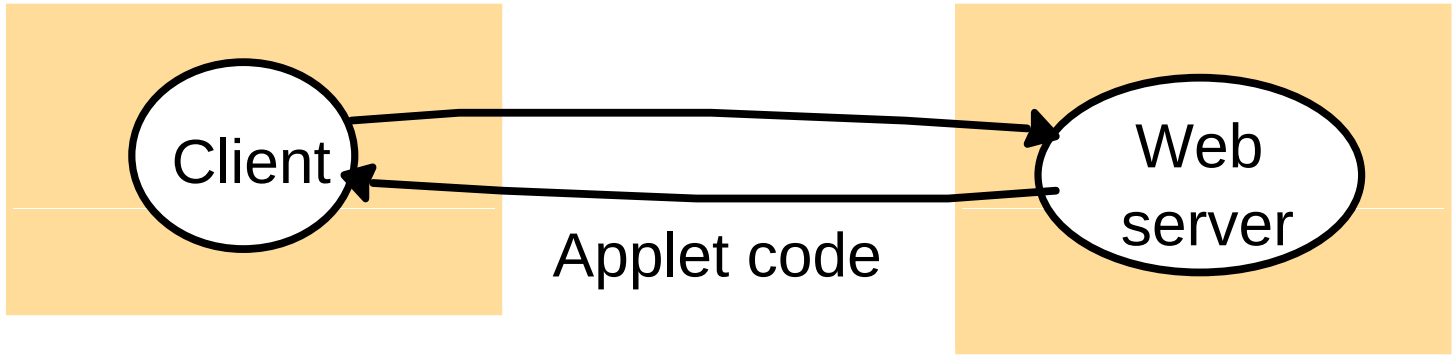



**Client/Server vs. peer-to-peer (P2P)**



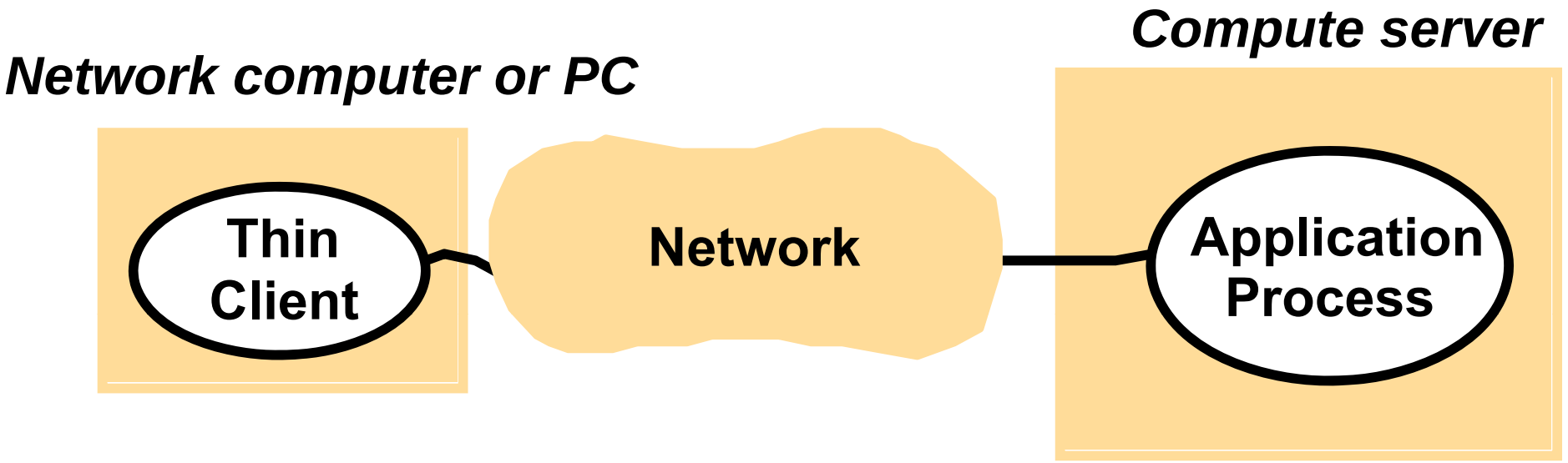
# Web applications: Web applets

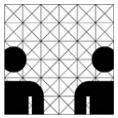
a) client request results in the downloading of applet code



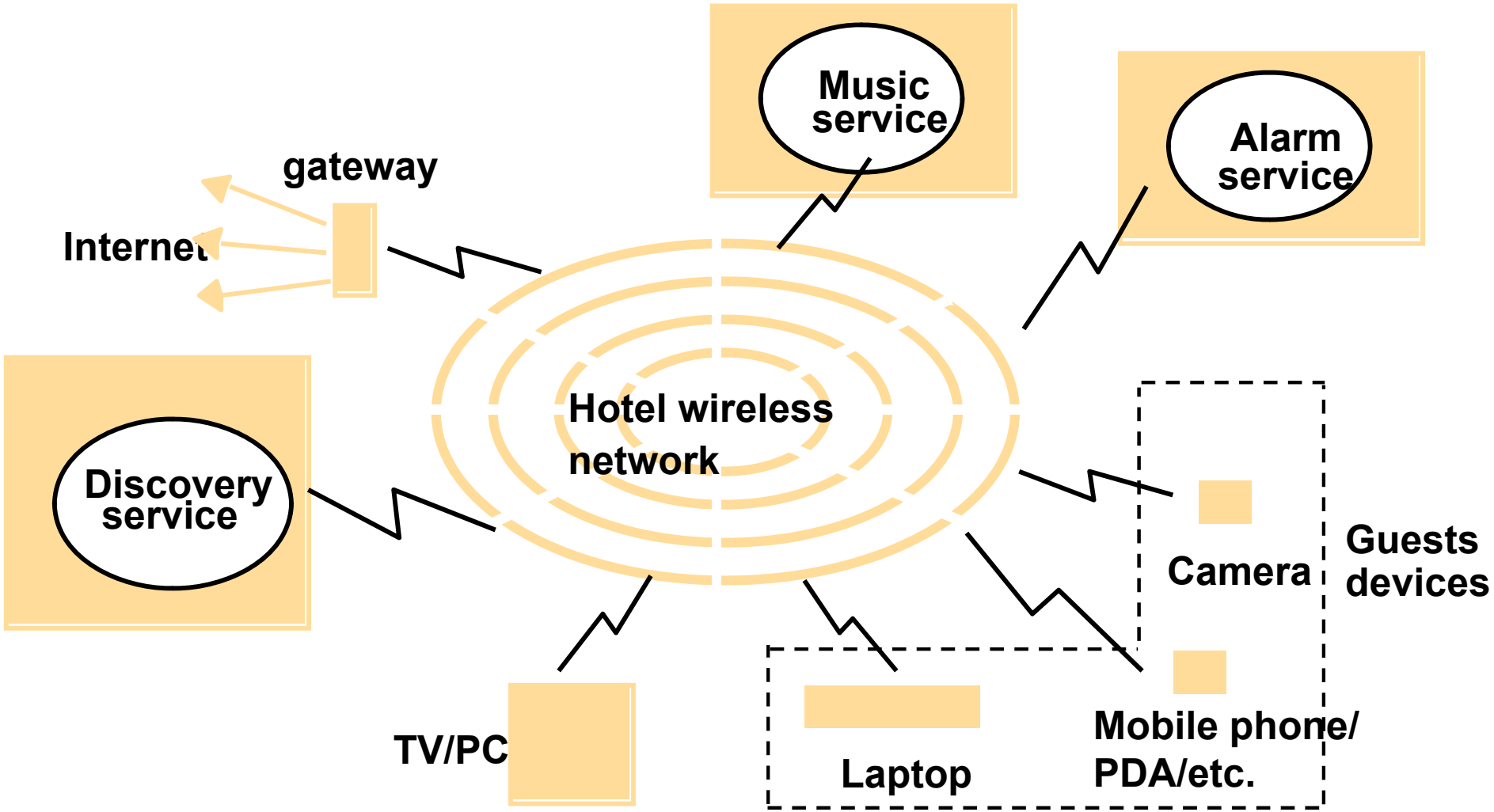

b) client interacts with the applet

# Client/Server: (d) Thin clients and compute servers



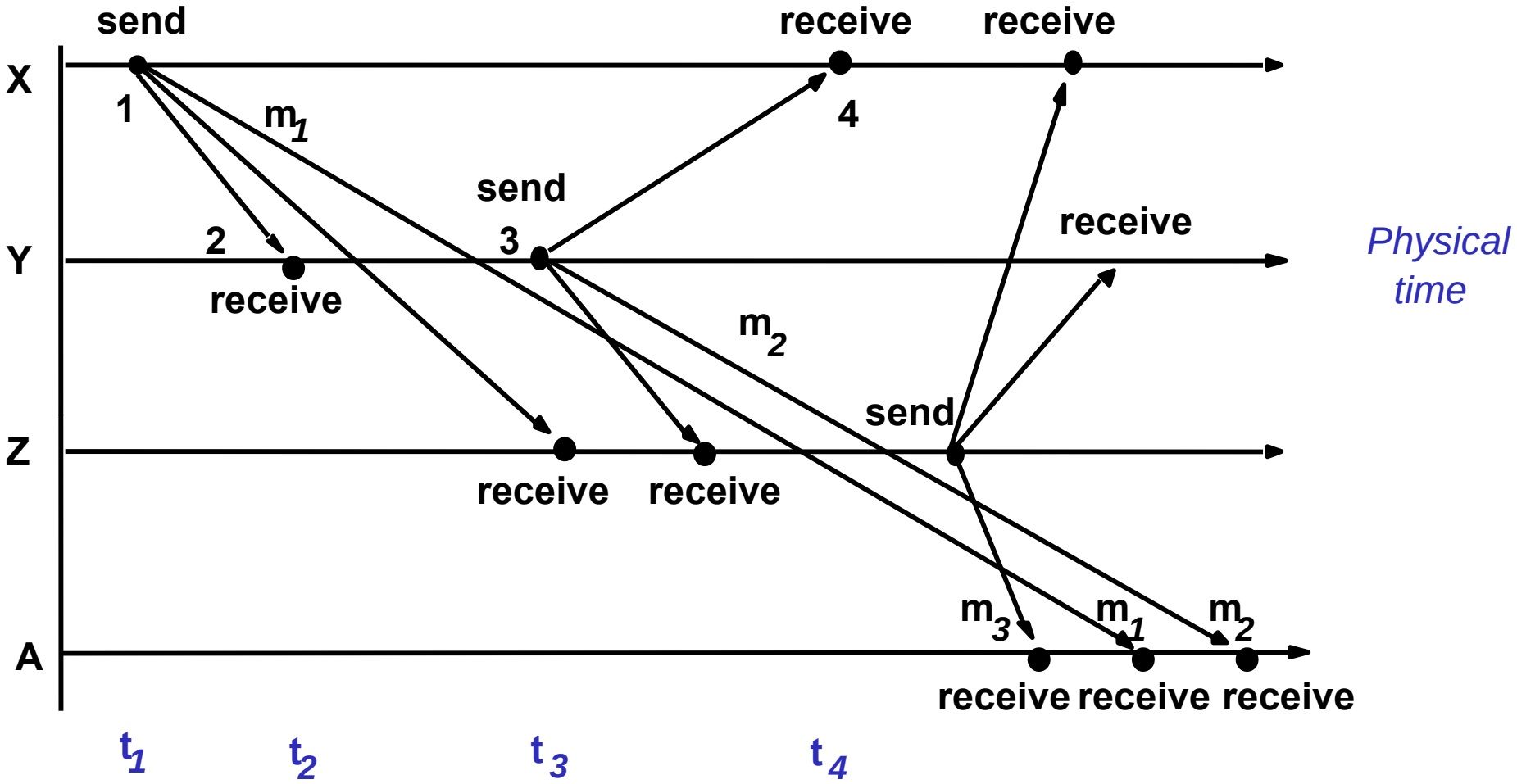



# Application example: Spontaneous networking in a hotel

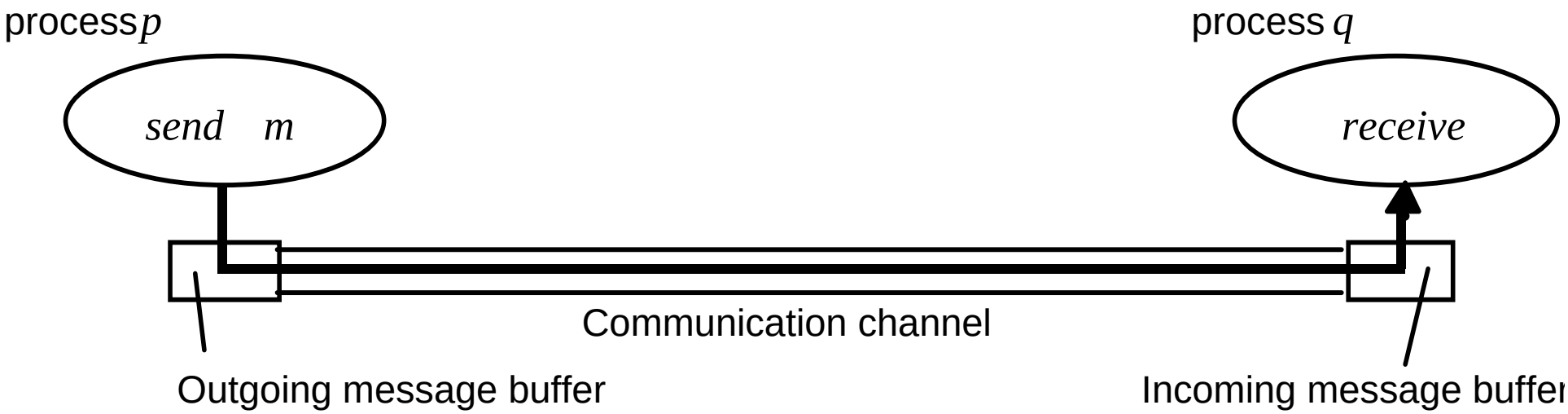


# Timing models: Real-time ordering of events

## *Asynchronous & synchronous* systems; *event ordering*

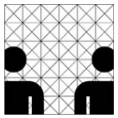


# Failure models: Processes and channels



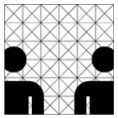

**2 types of failures:**

- **timing failures: in synchronous distributed systems**
- **arbitrary failures: (Byzantine failure) – „anything at any time“**



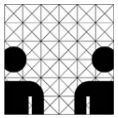
# Timing failures

| Class of Failure | Affects | Description |
|---|---|---|
| **Clock** | *Process* | Process's local clock exceeds the bounds on its rate of drift from real time. |
| **Performance** | *Process* | Process exceeds the bounds on the interval between two steps. |
| **Performance** | *Channel* | A message's transmission takes longer than the stated bound. |

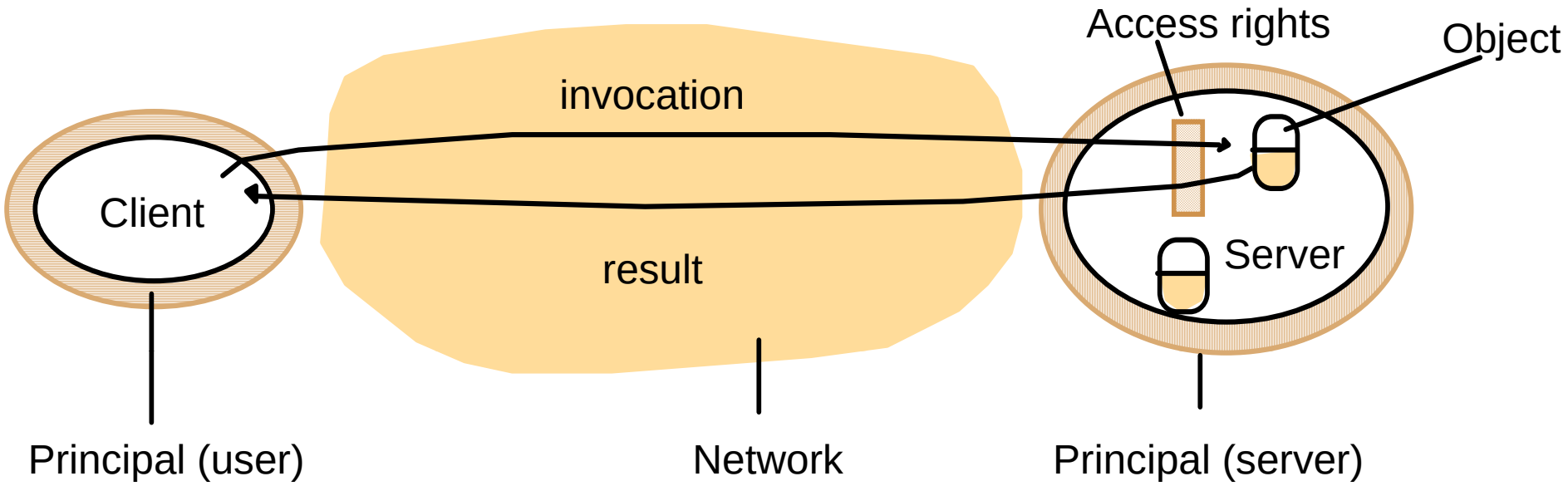


# Omission and arbitrary failures

| Class of failure | Affects | Description |
|---|---|---|
| **Fail-stop** | *Process* | Process halts and remains halted. Other processes may detect this state. |
| **Crash** | *Process* | Process halts and remains halted. Other processes may not be able to detect this state. |
| **Omission** | *Channel* | A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer. |
| **Send-omission** | *Process* | A process completes a *send,* but the message is not put in its outgoing message buffer. |
| **Receive-omission** | *Process* | A message is put in a process's incoming message buffer, but that process does not receive it. |
| **Arbitrary (Byzantine)** | *Process or channel* | Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step. |

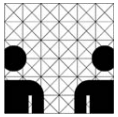


# Security models: Objects and principals

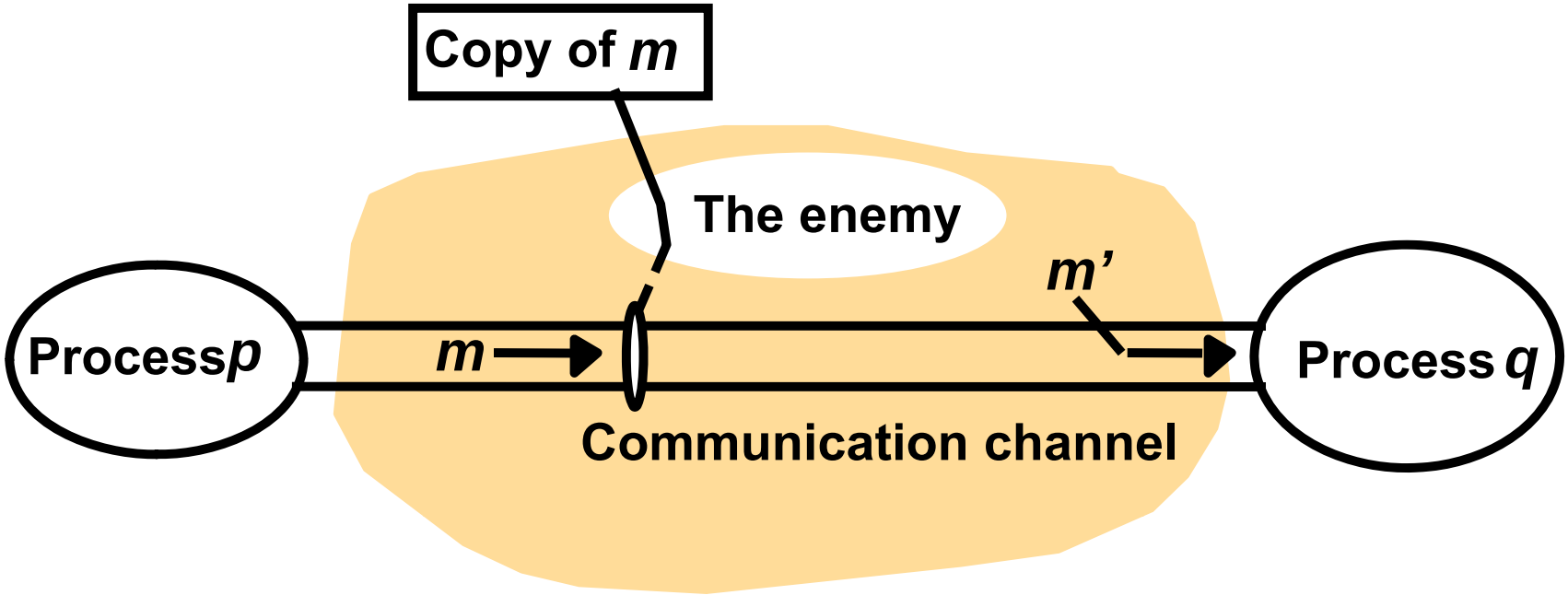



**Goals:**

- **Protecting objects**
- **Securing processes and their interaction**
- **Clients → "Agents" *(Principal-agent theory)***

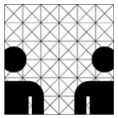


# Security models: The enemy

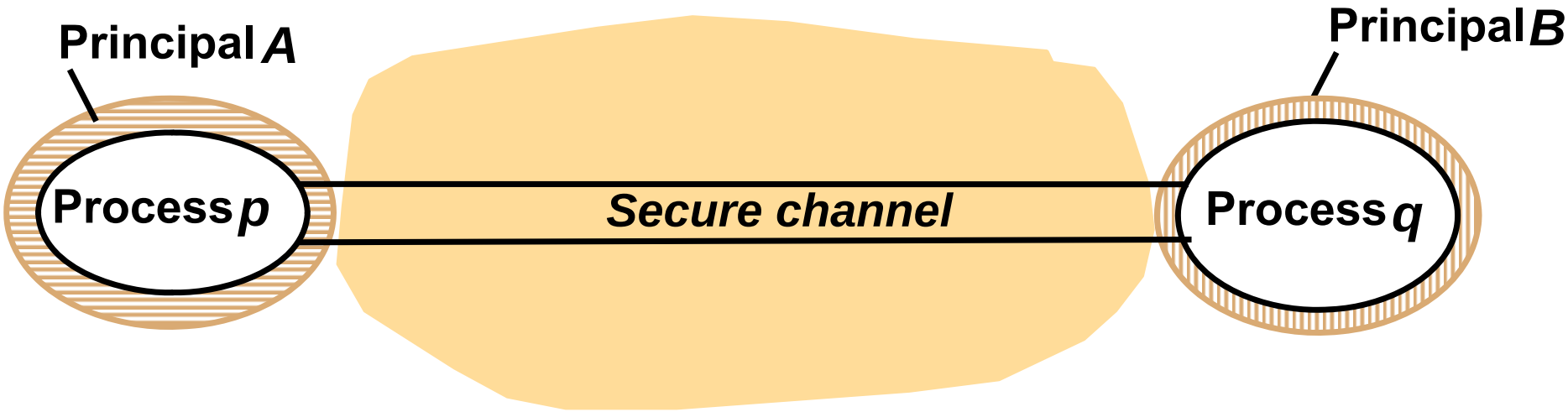



**Basis security threads:**

- loss of messages
- unauthorised read and/or access
- corruption
- false/stolen identities



# Security models: Secure channels




**Basis security techniques:**

- cryptography & shared secrets
- authorisation & authentication
- secure channels

→ **Sicherheitsaspekte: anderer Teil der Vorlesung GSS!**