TU Braunschweig
Institut für Betriebssysteme
und Rechnerverbund

Verteilte Systeme

Prof. Dr. Stefan Fischer

Kapitel 2: System-Modelle

Überblick

- Architektur-Modelle
  - Software-Ebenen
  - Client-Server
  - Mehrfache Server
  - Proxies
  - Peer-Prozesse
  - Varianten
  - Anforderungen an das Design
- Fundamentale Modelle für VS
  - Interaktionsmodell für die Komponenten
  - Fehlermodell
  - Sicherheitsmodell

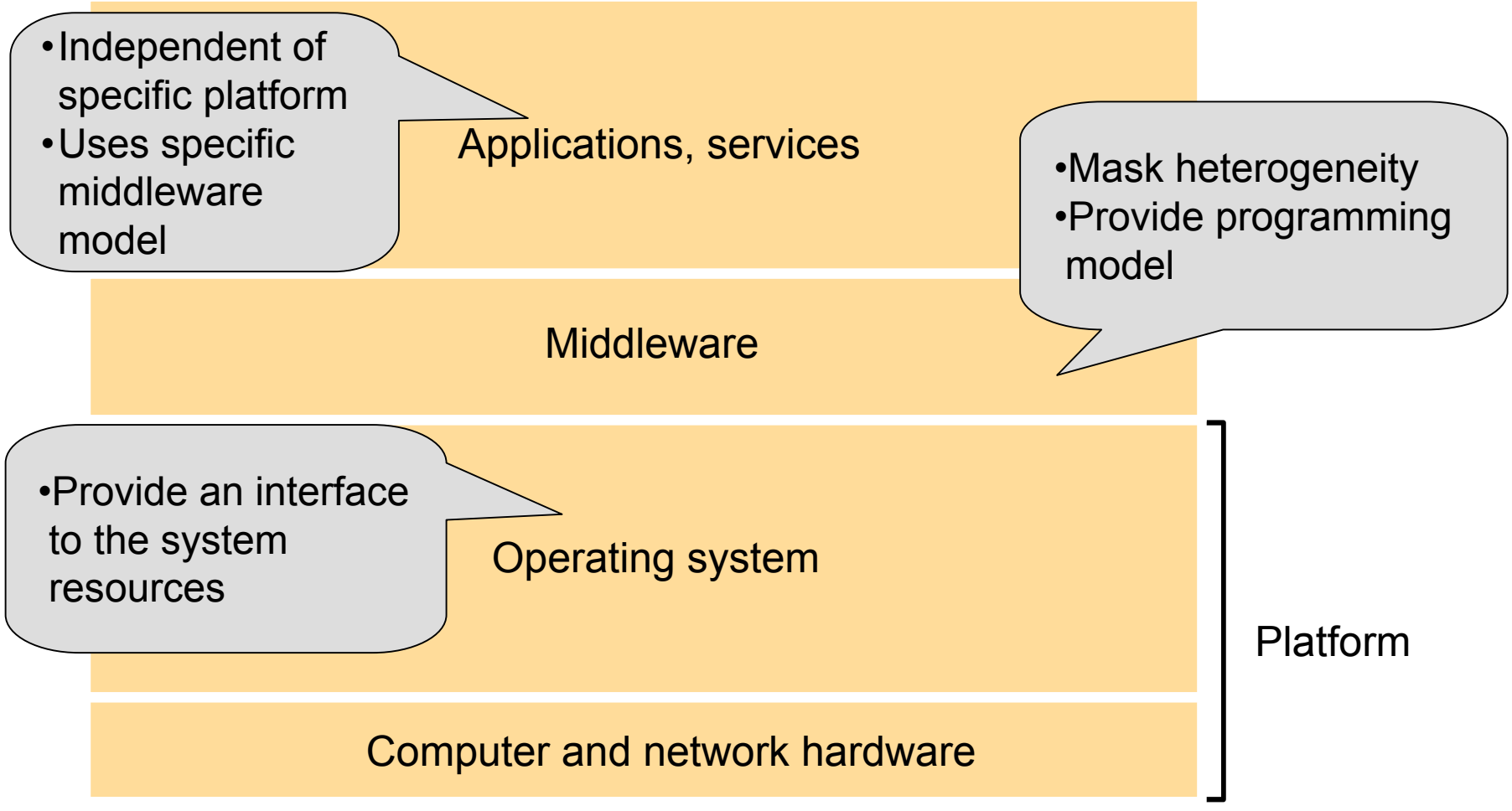
Der Zweck eines Modells

- Beschreibung der allgemeinen Eigenschaften und des Designs eines Systems
- Das Modell sollte abdecken:
  - Die wichtigsten Komponenten des Systems
  - Die Art ihrer Interaktion
  - Wie deren individuelles und kollektives Verhalten beeinflusst werden kann

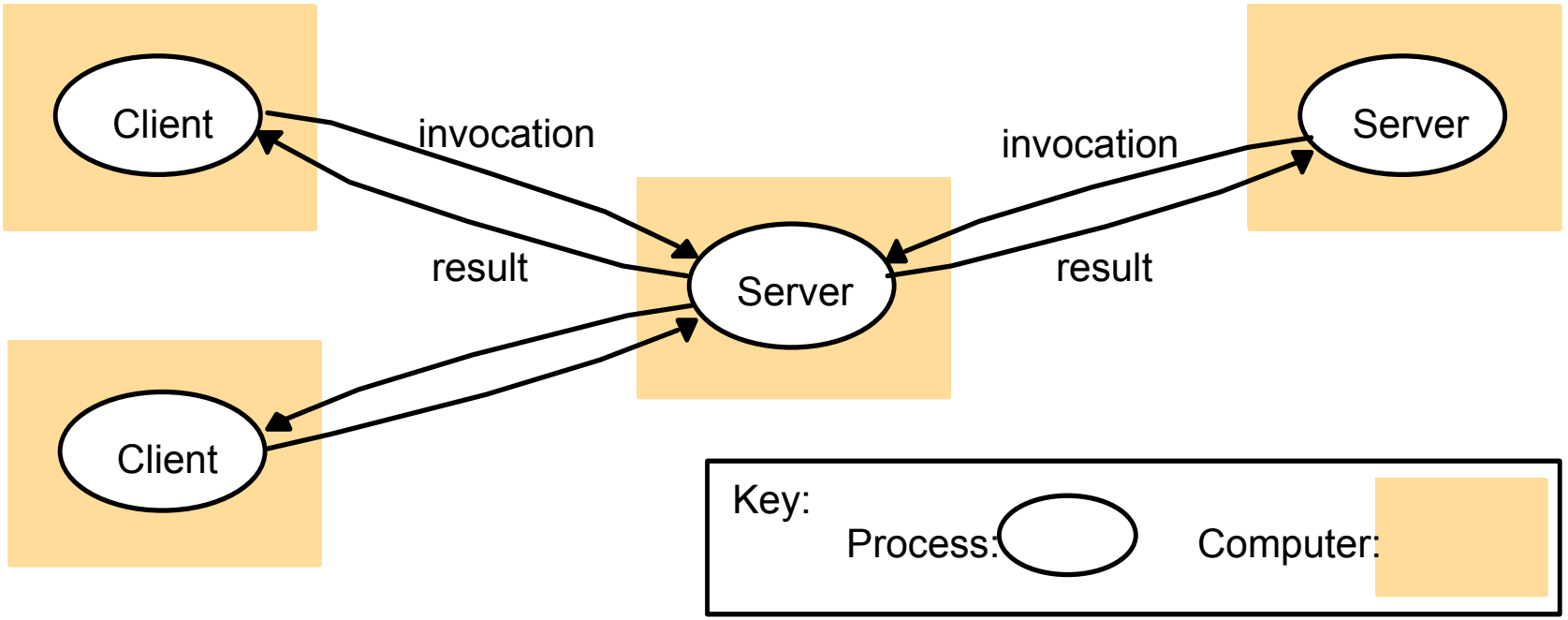
Architekturmodelle

- Ein Architekturmodell
  - vereinfacht und abstrahiert zunächst die Funktionen der individuellen Komponenten eines verteilten Systems, um dann
  - die Verteilung der Komponenten auf ein Netzwerk von Computern und
  - die Beziehung der Komponenten (Rolle in der Kommunikation mit anderen, Kommunikationsmuster) untereinander zu beschreiben.

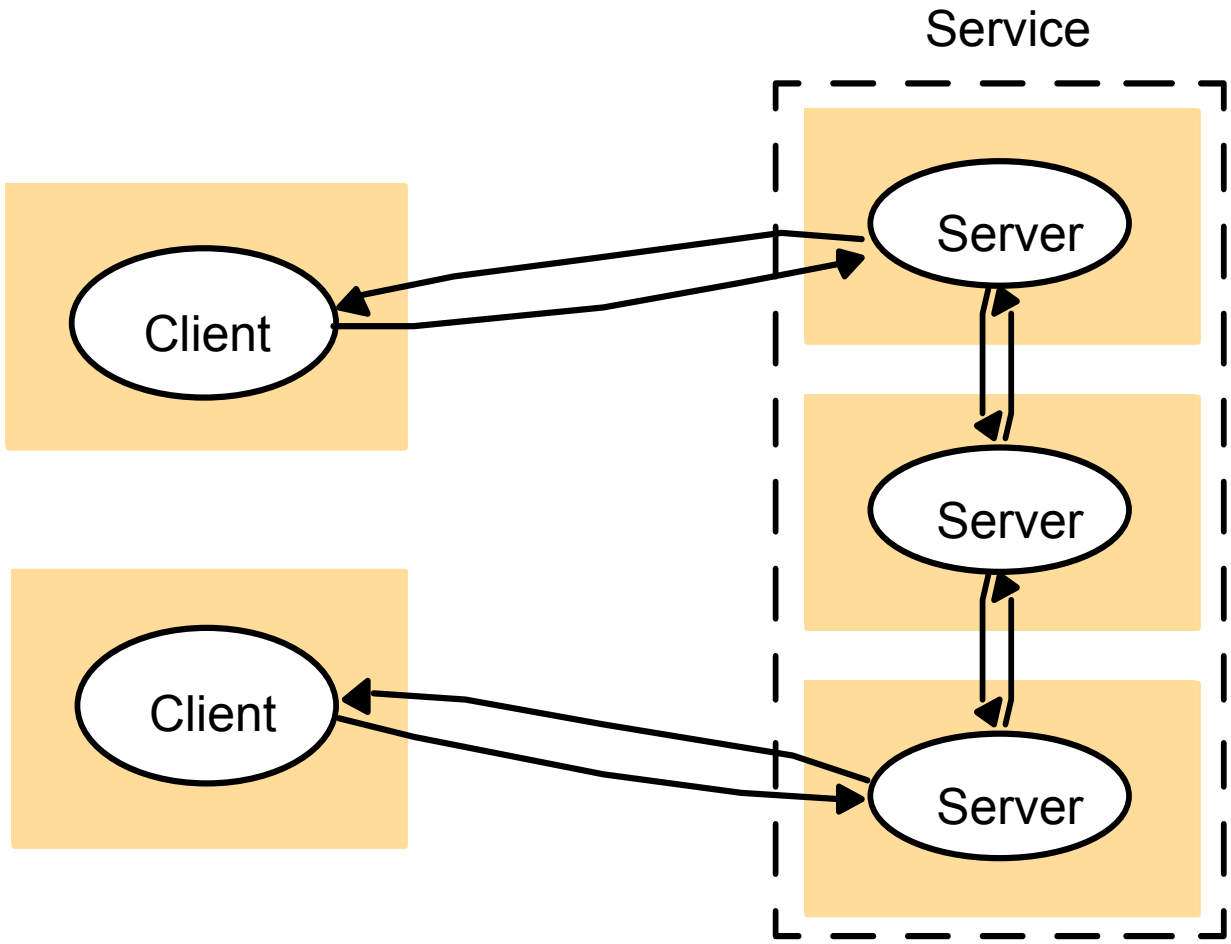
Software Layers



Client-Server



Mehrfache Server

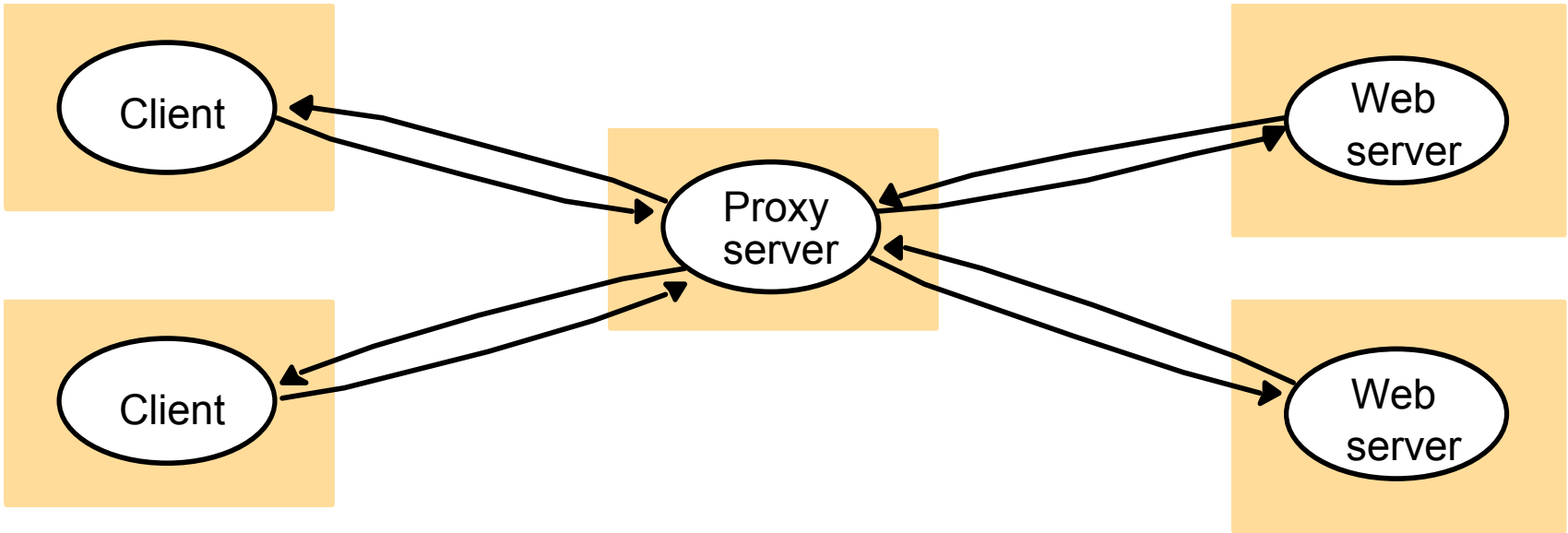


Partitition oder Replikation von Diensten

Beispiel Partition: WWW

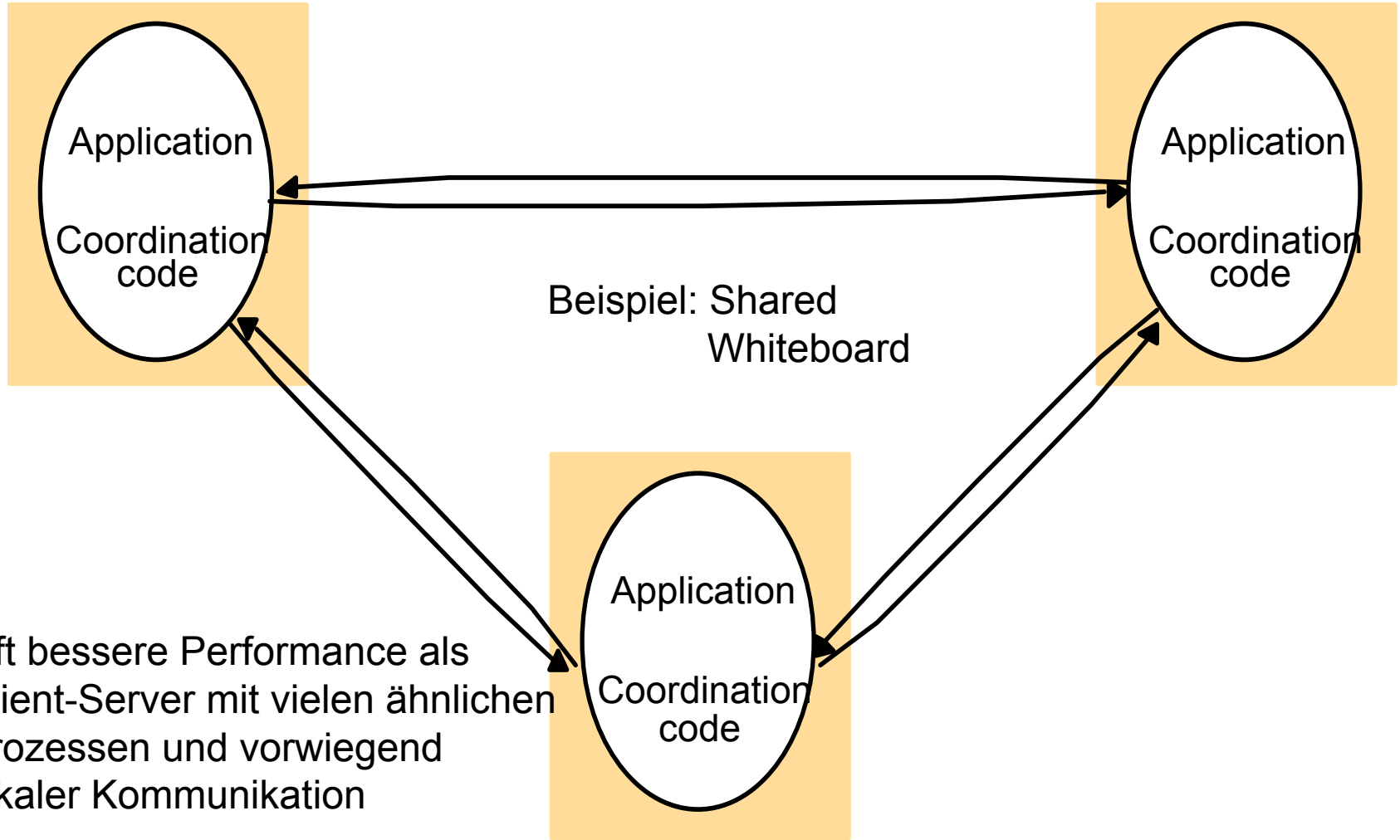
Beispiel für Replikation und Partition: DNS

Proxies



Zweck von Proxies: erhöhe Performance und Verfügbarkeit, erlaube sicheren Zugri

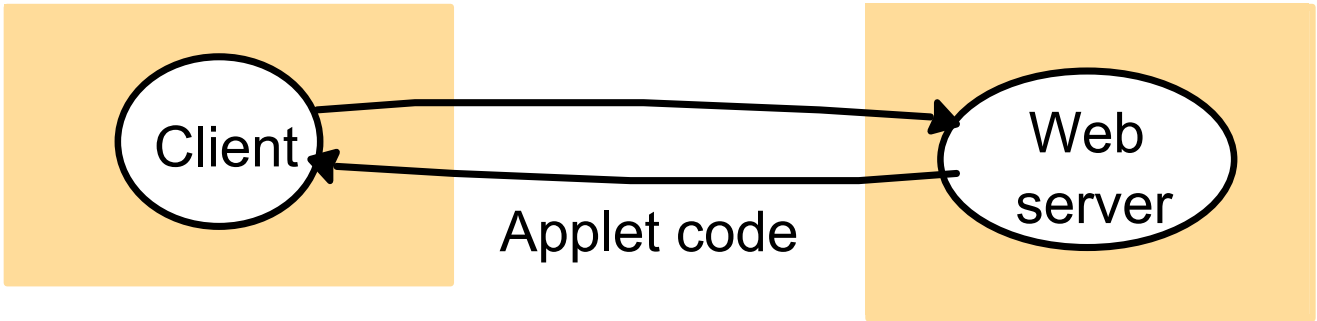
Peer-Prozesse



Oft bessere Performance als Client-Server mit vielen ähnlichen Prozessen und vorwiegend lokaler Kommunikation

Variante: Web Applets

a) client request results in the downloading of applet code

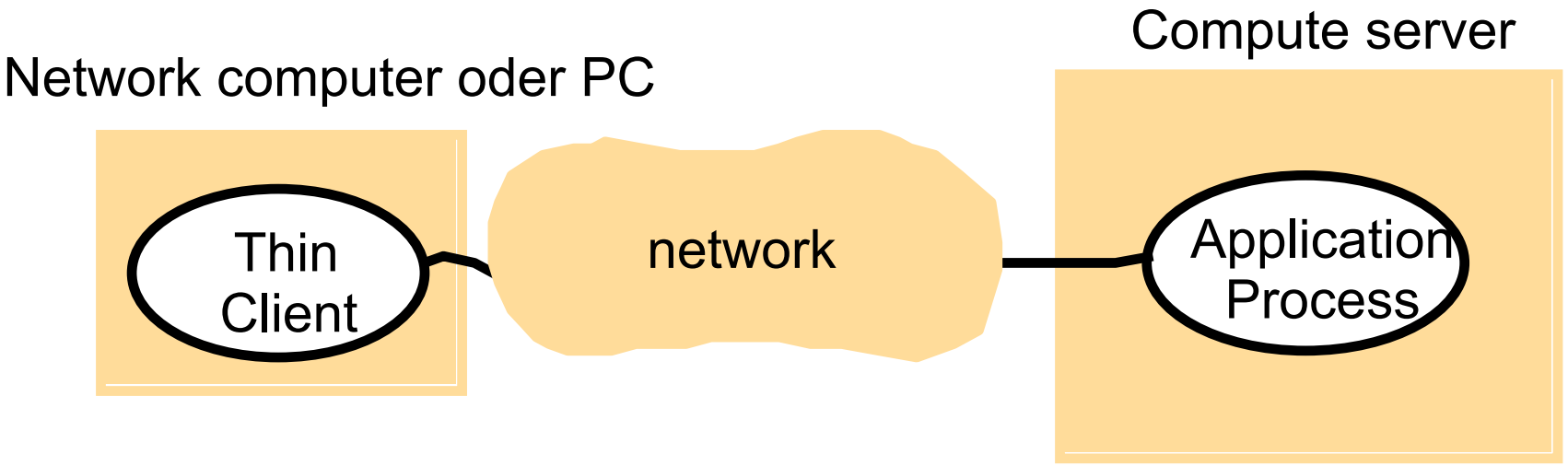


b) client interacts with the applet

Variante: Mobile Agenten

- Mobile Agenten sind Programmstücke, die von einem Server zum nächsten „reisen“, um
  - Informationen zu sammeln
  - Verhandlungen auszuführen
  - Produkte zu kaufen
  - und das alles im Auftrage eines Benutzers.
- Vorteil: Benutzer muss nicht selbst aktiv bleiben, kann sich sogar ausloggen und auf Ergebnisse warten
- Schwierigkeit: Implementierung guter Strategien

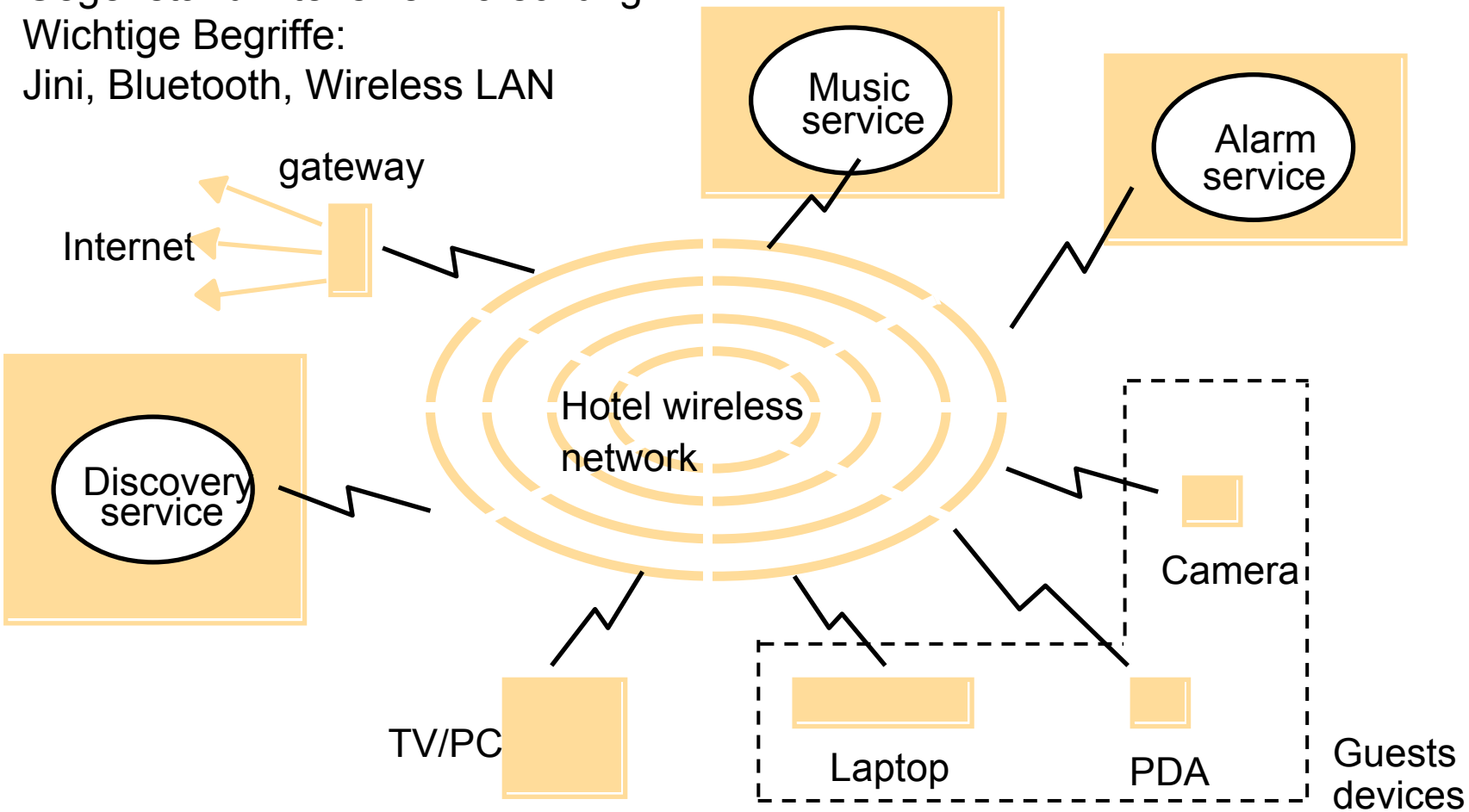
Variante: Thin clients



Dieses Modell wird von den Wettbewerbern von „Wintel“ favorisiert, wie etwa von Sun Microsystems. Man benötigt mächtigere Server und weniger mächtige PCs („network appliances“). Beispiel: Sunray von Sun.

Variante: Spontaneous Networking

Dieses Modell ist immer noch
Gegenstand intensiver Forschung.
Wichtige Begriffe:
Jini, Bluetooth, Wireless LAN



Anforderungen an die Auswahl eines Modells

- Welches sind die Einflussfaktoren
  - bei der Auswahl des Modells
  - Bei der Platzierung der Komponenten

- Wichtig
  - Performance
  - Quality of Service
  - Inwieweit sollen Caching und Replikation eingesetzt werden
  - Benötigter Grad der Verlässlichkeit des Systems
  - Kosten

Performance

- Benutzeranforderungen an die Leistungsfähigkeit eines Systems:
  - Antwortverhalten: Benutzer verlangen schnelle und konsistente Antworten. Dazu benötigt man
    - Wenige Komponenten
    - Lokale Kommunikation, wo möglich
    - Austausch kleinstmöglicher Dateneinheiten
  - Durchsatz: Rate, mit der die Berechnungen durchgeführt werden. Die schwächste Komponente bestimmt den Durchsatz.
  - Lastbalancierung: Verschiebung von Arbeitsteilen auf weniger belastete Komponenten. Replikation ist notwendig.

Quality of Service

- Wenn erst einmal die Funktionalität eines Dienstes gewährleistet ist, wird schnell die Frage der Dienstgüte aufkommen.
- Zu bedenkende Fragen
  - Zuverlässigkeit, Verlässlichkeit
  - Sicherheit
  - Mehr und mehr echtzeit-orientierte Dienstgüteparameter (Beispiele?)

Caching und Replikation

- Wenn Caching oder Replikation eingesetzt werden, muss für viele Anwendungen sichergestellt werden, dass die Benutzer immer die aktuellste Kopie bekommen.
- Wir betrachten das Thema in Kapitel 11.

Verlässlichkeit

- Verlässlichkeit („dependability“) umfasst
  - Korrektheit: Das System sollte das erwartete Verhalten zeigen, was z.B. in der Spezifikation festgelegt ist.
  - Sicherheit: Welches ist z.B. der sicherste Ablageplatz für Daten?
  - Fehlertoleranz: beschreibt, inwieweit ein Dienst, ein System immer noch arbeitet, selbst wenn Fehler auftreten

Beispiele

- Web Server der Familie Fischer?
- Web Server von Google?
- Banken-Informationssysteme?

Fundamentale Modelle

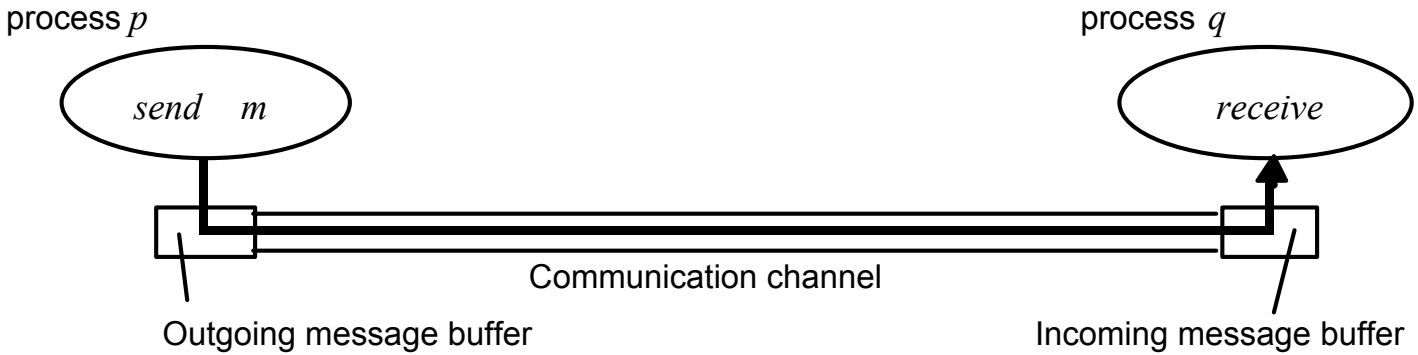
- Neben der Architektur eines Systems müssen weitere, fundamentalere Fragen bedacht werden, die jedem System gemein sind:
  - Art der Interaktion zwischen den Komponenten
  - Fehler in einem System und deren Behandlung
  - Sicherheitsfragen

Interaktionsmodelle

- Berechnungen erfolgen in Prozessen
- Prozesse interagieren durch Nachrichtenaustausch
  - Zur Kommunikation (Datenübermittlung)
  - Zur Koordination (Synchronisation, Ordnung von Ereignissen)
- Das Interaktionsmodell muss vor allem die *Verzögerung* durch Netzkommunikation in Betracht ziehen.
- Die Genauigkeit der Synchronisation wird durch diese Verzögerung beschränkt sowie durch das Fehlen einer echt globalen Zeit im VS.

Leistung des Kommunikationskanals

- **Latenz**: Verzögerung zwischen dem Senden einer Nachricht durch den einen und deren Empfang durch den anderen Partner; umfasst
  - Die reine Übertragungszeit des Mediums
  - Die Verzögerung beim Zugriff auf das Medium
  - Die lokale Verarbeitungszeit im Betriebssystem
- **Bandbreite** des Netzwerks, die evtl. mit anderen Prozessen geteilt werden muss. Bestimmt den möglichen Durchsatz
- **Jitter**: Varianz in der Latenz. Wichtig für Echtzeitanwendungen.



Zeit in VS

- Jeder Computer in einem VS besitzt seine eigene lokale Uhr.
- Man kann generell nicht davon ausgehen, dass die Uhren die gleiche Zeit verwenden.
- Für viele Anwendungen ist es jedoch wichtig festzustellen, wer zuerst eine Aktion ausführt – Beispiele?
- Möglichkeiten
  - Physikalische (teure) Lösungen
  - Protokollösungen, die nicht so genau sind, aber oft genügen

Fehlermodelle

- In einem VS können sowohl in Prozessen als auch Kommunikationskanäle Fehler auftreten.
- Fehler = Abweichen vom korrekten oder wünschenswerten Verhalten
- Arten von Fehlern:
  - Unterlassungsfehler (omission)
  - Zeitfehler (timing)
  - willkürliche Fehler (arbitrary)
- Generell können Fehler maskiert werden, so dass andere Komponenten unbeeinflusst weiterarbeiten können.

Unterlassungs- und willkürl. Fehler

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.

Timing-Fehler

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.

Sicherheitsmodelle

- Die Sicherheit eines VS kann erreicht werden, indem die Prozesse und Kanäle gesichert und die Objekte, die von Prozessen verwaltet werden, gegen unautorisierten Zugriff geschützt werden.
- Zur Definition des „unautorisierten Zugriffs“ muss der Benutzer in das Modell aufgenommen werden.
- Außerdem muss man über „Angreifer“ oder „Feinde“ reden können sowie über deren Angriffsarten.
- Schließlich müssen Lösungsmöglichkeiten gefunden werden.

Angriffspunkte