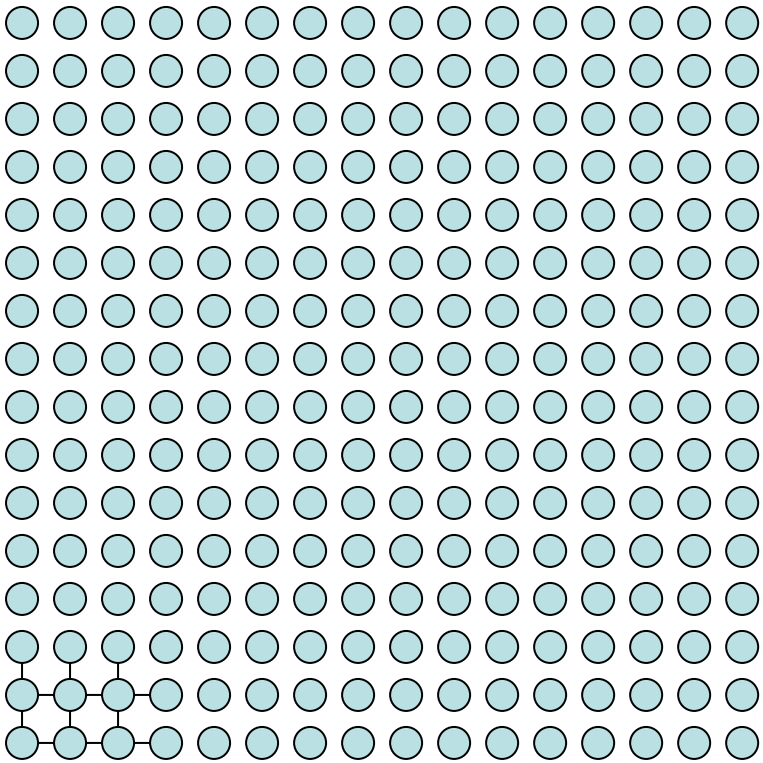
Verteilte Systeme - Übung

- Schriftliche Übungen
- Dienen der Klausurvorbereitung
- Zwei Teile:
  - Serie A: Okt - Nov
  - Serie B: Nov - Jan
- 3% der Endnote je Serie
- Ansprechpartner: Harald Vogt **<vogt@inf.ethz.ch>**

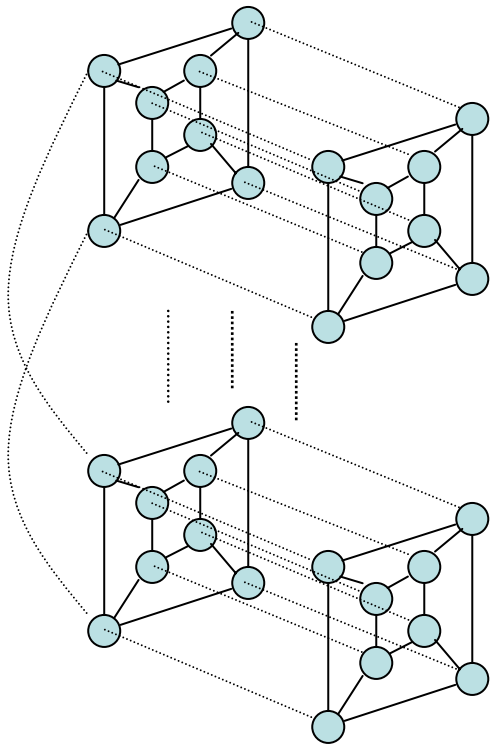
Heute:

- Vorbesprechung der Aufgaben Serie A
- Echo-Algorithmus
- Fragestunde

A1. Topologien



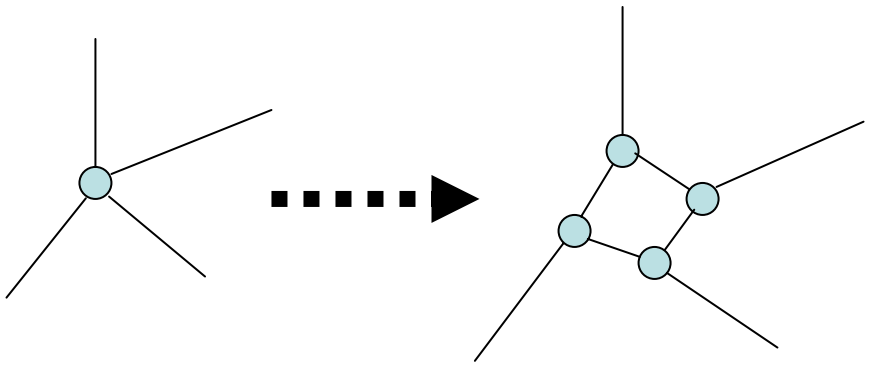
Gitter, 16x16



Hypercube, d=5

Maximale Pfadlängen?

A1. Topologien



Ecke eines Hypercube, d=4

Ecke eines Cube Connected Cycle

Anzahl der Knoten im CCC?
Maximale Pfadlänge im CCC?
(Achtung!)

A2/A3. Pfade im Hypercube

- Anzahl der Pfade im Hypercube
- Anzahl der **disjunkten** Pfade im Hypercube
  - Disjunkte Pfade: keine gemeinsamen Knoten ausser Anfang und Ende
  - Wieviele disjunkte Pfade kann es maximal geben?
- Mittlere Pfadlänge im Hypercube
  - Hypercube-Konstruktion beachten
  - Beweis-Skizze

A4. Fehlermodelle

- Kommunikation ist fehlerbehaftet
  - Fehler beim Senden, Empfangen, Übertragen
  - Crash (Fehler nicht erkannt)
  - Fail-Stop (Erkennung des Fehlers)
  - Zeitfehler
  - Byzantinisch (beliebiges Verhalten des Hosts/Kanals)
- Es muss klar sein, wer/was der Sender/Empfänger einer Nachricht ist
  - Z.B. Endbenutzer, Mail-Client, Web-Server, etc.
- Manchmal nicht ganz eindeutig...

A5. Fehlertoleranz

- Was zum Rechnen...
- 60% Verfügbarkeit
  - Schlecht, oder?
  - Der **Heilige Gral**: „Five nines“ 99.999% - 5 min 15 sec downtime/year
  - „They learned that 65 percent of system reboots were due to ***planned*** outages for routine **administrative tasks** such as adding hardware and applications. Of the ***unplanned*** outages, 21 percent were caused by **application failures**, and 14 percent were due to **system failures**. More than half of the system failures were traced to **device drivers, anti-virus software, and hardware failures**. (Note: This finding supports industry studies that say as much as 80 percent of system failures can be traced to errors caused by **people** or **flawed processes**, an issue addressed in the People and Processes section below.)“*
- Erhöhung der Verfügbarkeit durch Redundanz
  - Mit 3 Rechnern
  - Wieviele Rechner für 99%?
  - Ausfall in Stunden/Jahr

*(Source: http://www.microsoft.com/windows2000/server/evaluation/business/overview/reliable/default.asp)

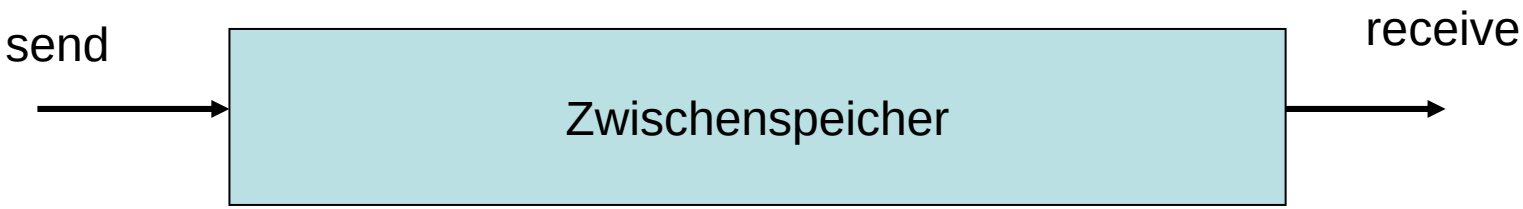
A6. Puffer-Implementierung

- Java-Klasse `SyncPort`:
  - `public synchronized void send(int n)`
  - `public synchronized int receive()`

Nachrichten sind vom Typ `int`

- Synchrone Kommunikation: „gleichzeitig“, d.h. es wird ein definierter Punkt im Programmablauf erreicht
  - Wichtig z.B. bei der Benutzerführung (Fortschrittsanzeige)
  - Kommunikation zwischen 2 **Threads**
  - `Send` blockiert, bis `receive` aufgerufen wird
  - `Receive` blockiert, bis `send` aufgerufen wird
- „Unnatürlich“ auf „normalem“ Rechner
  - 1 Prozessor: immer nur 1 Thread aktiv
  - Mehrere Threads: Scheduling verursacht Verschiebungen
  - Hauptsächlich asynchrone Primitive in Linux/Windows:
    - Sockets, Pipes
  - Aber auch synchrone Mechanismen: MPI

A6. Puffer



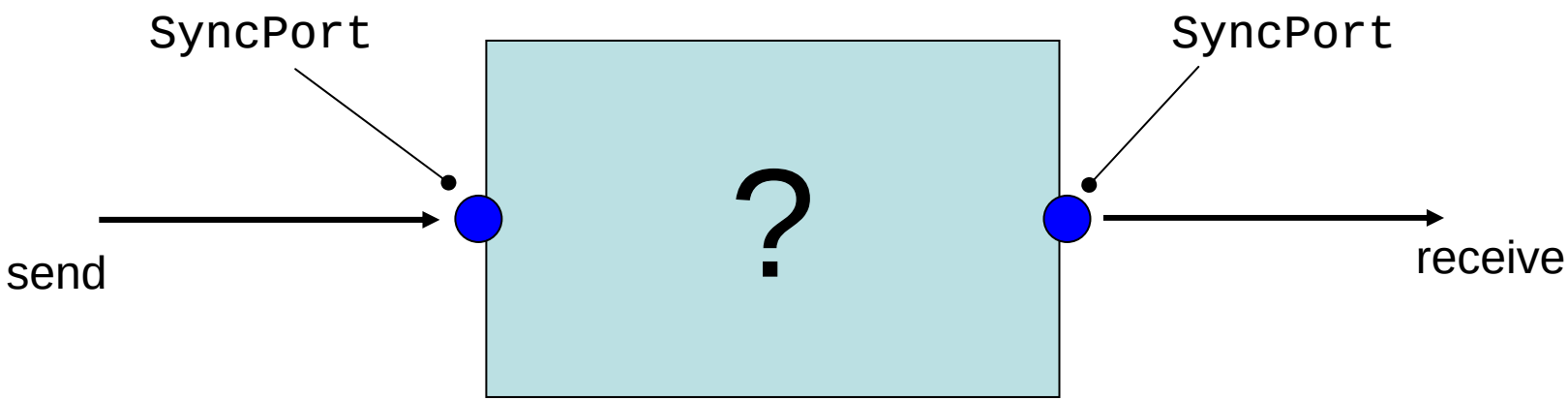
Aufruf kehrt sofort zurück,
falls Platz im Zwischenspeicher

Aufruf kehrt sofort zurück,
falls eine Nachricht im Zwischenspeicher

- Gemeinsamer Synchronisationspunkt der Programme geht verloren!
- Dafür Ausgleich von „Bursts“
- Kann keine dauerhafte Ungleichheit ausgleichen!

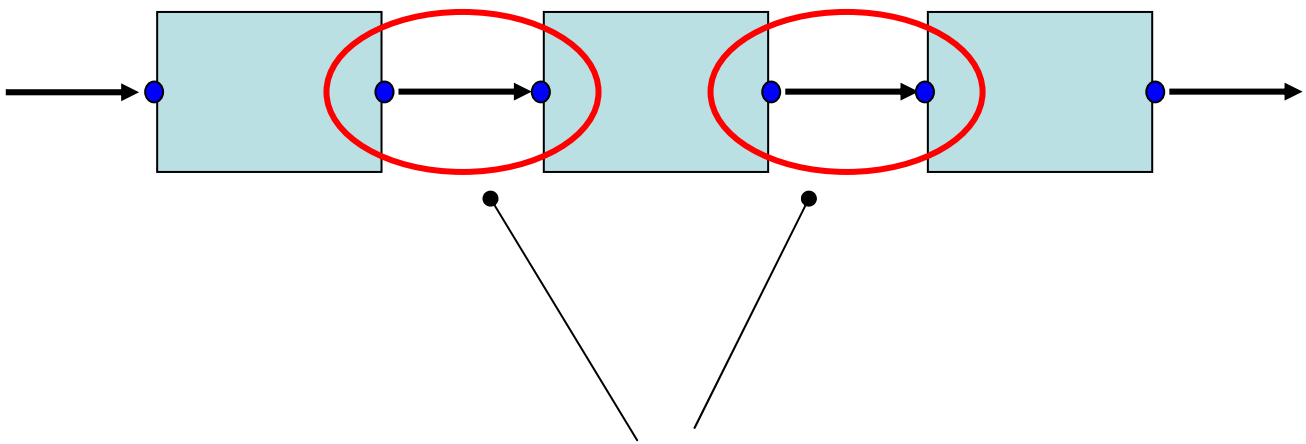
A6. Puffer

Puffer für 1 Element:



A6. Puffer

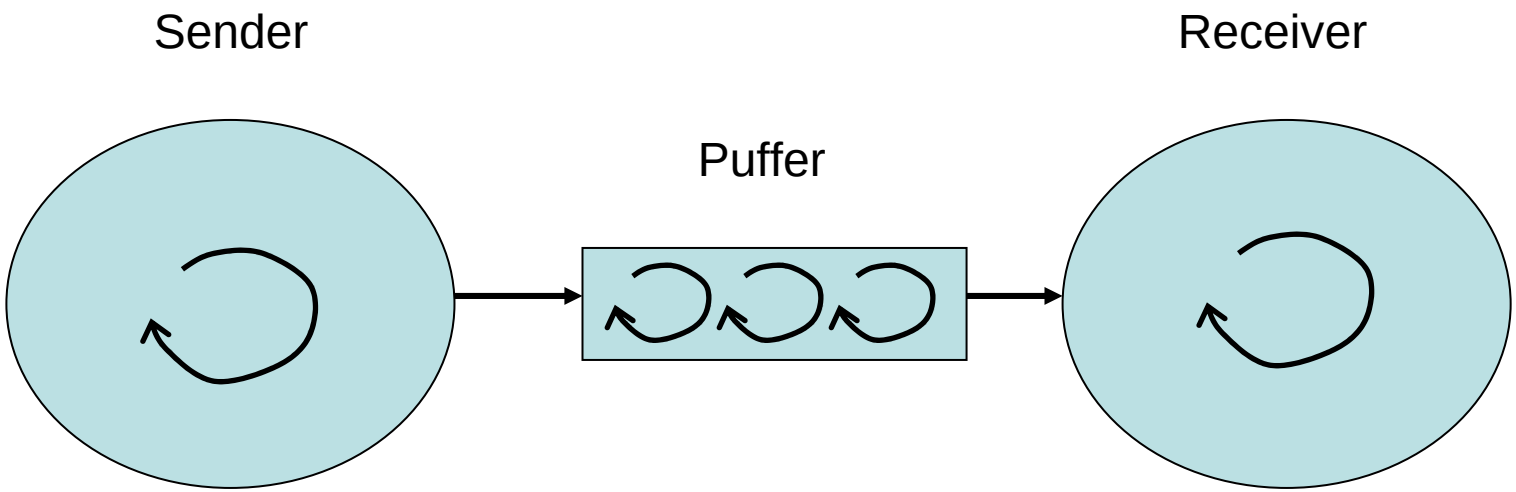
Puffer für 3 Elemente:



Vielleicht besser mit nur einem SyncPort an diesen Stellen?

A6. Puffer

Testaufbau:



Empfänger liest mit gleichförmiger Rate. Sender ändert seine Senderate von Zeit zu Zeit. Puffer kann also voll/leer werden, was zu Wartezeiten führt.
Wie lange müssen Sender u. Empfänger für einen Testlauf insgesamt warten?
Kommen alle gesendeten Nachrichten beim Empfänger an?

A7. Mars-Rover

- Was zum Rechnen...
- Wie schnell fährt ein Mars-Rover?
- Kann man mit ihm „synchron“ kommunizieren?

A8. RPC

- Referenztypen bei RPC?
  - Referenz = lokale Adresse
- Timeout und Wiederholung von RPC-Anfragen
  - Gefahr? Gegenmassnahme?
- RPC-Fehlersemantik
  - Maybe, At-least-once, At-most-once, Exactly-once
  - Anwendung auf Web

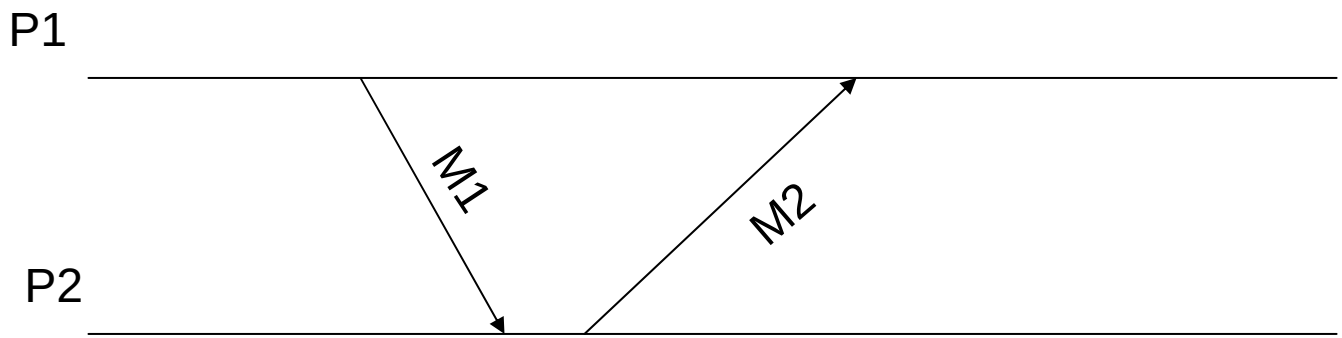
A9. Broadcast

- Atomar = total geordnet
  - Wenn Pi die Nachrichten M1, M2 in dieser Reihenfolge erhält, dann auch Pj
- Kausale Abhängigkeit
  - Es gibt einen Pfad vom Sendeereignis M1 zum Sendeereignis M2...
- Hilft zentraler Sequencer für Einhalten der totalen Ordnung?

A10. Uhrensynchronisation I

- Unterschiedliche Geschwindigkeit von Uhren
- Periodische Synchronisation
- Was zum Rechnen...

A11. Uhrensynchronisation II



- Uhrensynchronisation bei variabler Nachrichtenlaufzeit
- Welche Genauigkeit kann man erreichen?
- Ereignisse tragen Zeitstempel!
- Zeitstempel kann in Nachricht versendet werden
- Verzögerung bei P2 zwischen Empfangs- und Sendeereignis irrelevant