Architektur verteilter Anwendungen

- Schwerpunkt: verteilte Algorithmen
- Algorithmus: endliche Folge von Zuständen
- Verteilt:
  - unabhängige Prozessoren rechnen
  - tauschen Informationen über Nachrichten aus
- Komplexität: Zeitverbrauch+ #Nachrichten


in diesem Modell sind alle Prozesse unabhängig
und kommunizieren miteinander

Architektur verteilter Anwendungen

- **Vorlesungsteil:**
  Ideen und Konzepte
  für algorithmische Probleme
  in verteilten Systemen

- **Praktischer Teil:**
  Implementierung dieser Konzepte
  zur Erkennung der Problematik

- **Seminarteil:** Vorstellen eigener Implementierung

Modelle für verteilte Anwendungen

- sehr individuell, Kurse heißen
  - Verteilte Systeme
  - Verteilte Anwendungen
  - Verteilte Algorithmen
  - Nicht-sequentielle Programme
  - ...

Architektur verteilter Anwendungen

- Vorlesung mit 3 integrierten Projektaufgaben
  Lehrform 1V+1S+2P
- Anmeldung per e-Mail: Name+Matnr.

- Webseite für Veranstaltung
  `www-crypto.htwsaar.de/weber/teaching/15_ws_ava/`
  - Übungsblätter,
  - Status der Bewertung,
  - Abnahmeplan

Architektur verteilter Anwendungen

- Prüfungsleistung:
  - praktische Übungen (50%)
  - mündliche Prüfung (50%)
- Bonuspunkte für die mündliche Prüfung
  - 30 min Seminarvortrag eigener Lösung: 10%
  - max 3 eigene Vorträge (jew. Dienstag)

ISL 6226 (Technikum)

ISL 6226 (Technikum)

- Treffpunkt Mi/3. für Fragen, Betreuung, Abnahme

  BYOD = bring your own device or use ISL

- STL-Account, STL-Homepartition, ssh
  C/C++/Python/Ruby
- `isl-s-01`
  `Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz`
- `isl-c-01 ... isl-c-10`
  `Intel(R) Core(TM) i5-3570 CPU @ 3.40GHz`

Projektaufgaben / Übungen

htw saar

Studiengang Kommunikationsinformatik (Master)
Studiengang Praktische Informatik (Master)
Prof. Dr.–Ing. Damian Weber
Sarah Theobald, M.Sc.

Architektur verteilter Anwendungen – Übung 1

Aufgabe 1 (Basisimplementierung eines lokalen Knotens)

Implementieren Sie einen lokalen Knoten als einen Prozess, d.h. *kein Thread*. Der Knoten soll folgendes tun:

1) als Eingabe eine Datei lesen, die zu *ID*s Endpunkte als IP–Adresse und Portnummer zuordnet

```
Beispiel:
1 isl-s-01:5000
2 isl-s-01:5001
3 127.0.0.1:2712
...
```

Praktische Aufgaben zu den besprochenen Konzepten/Algorithmen

Projektaufgaben / Übungen

Dokumentieren Sie Ihre Arbeitsschritte und Ergebnisse !

Projektaufgaben / Übungen

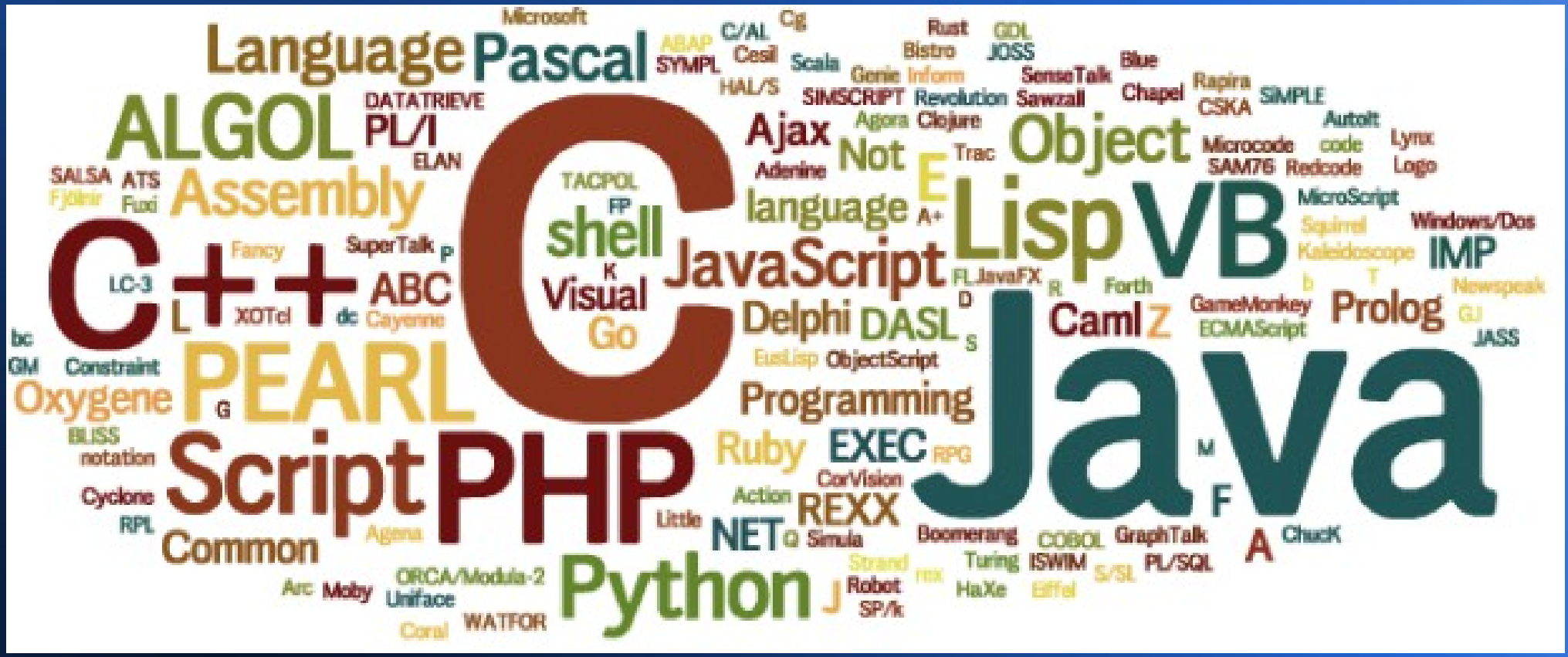
- Übungen müssen vor Abgabe besprochen sein (Abnahme)

- Lösungen enthalten ein README (.txt, .doc, .odt)
  - Erläuterung der Idee
  - Nachrichtenformat (frei, JSON, XML, YAML, etwas Neues)
  - Erläuterung der Softwarestruktur
  - Besonderheiten Implementierung (Stärken, Schwächen)
  - typische Beispielabläufe
  - Fazit (gewonnene Erkenntnisse)
- Programmcode kommentiert

Letzte Abnahmemöglichkeit: Mittwoch 27.01.2016

BYOD

BYOL


Microsoft
Language
Pascal
ABAP
C/AL
Cg
Rust
GDL
SYMPL
Cesil
Scala
Bistro
JOSS
Blue
HAL/S
Genie
Inform
SenseTalk
Rapira
SIMSCRIPT
Revolution
Sawzall
Chapel
CSKA
SIMPLE
ALGOL
DATATRIEVE
PL/I
Agora
Clojure
Ajax
Not
Object
Autolt
Microcode
code
Lynx
ELAN
Trac
SAM76
Redcode
Logo
SALSA
ATS
Adenine
Assembly
TACPOL
FP
shell
language
E
A+
Lisp
VB
MicroScript
Squirrel
Windows/Dos
Fjölnir
Fuxi
C
Fancy
SuperTalk
P
K
Visual
JavaScript
FLJavaFX
Kaleidoscope
IMP
LC-3
C++
ABC
L
XOTcl
dc
Cayenne
Go
Delphi
DASL
D
R
Forth
Caml
Z
GameMonkey
ECMAScript
Prolog
b
T
Newspeak
GJ
bc
JASS
GM
Constraint
S
EusLisp
ObjectScript
Oxygene
PEARL
G
Programming
Java
BLISS
notation
Ruby
EXEC
RPG
M
Cyclone
CorVision
Action
REXX
F
RPL
Script
PHP
Little
NET
Q
Simula
Boomerang
COBOL
GraphTalk
A
ChucK
Common
Agena
Strand
Turing
ISWIM
PL/SQL
Arc
Moby
ORCA/Modula-2
Uniface
Python
J
Robot
SP/k
rex
HaXe
Eiffel
S/SL
Coral
WATFOR

Programmcode

- lernen Sie evtl. eine neue Sprache
  - Python # aber Kommentare benutzen
  - Ruby # keine Scheu vor Anmerkungen
  - C# // kommentieren wie in Java
  - ...
- übersichtlich, modular, selbsterklärend

Abnahmeplan

- Abnahme Mittwoch (später auch Dienstag)
- Anmeldung per e-Mail
- aktualisierter Abnahmeplan auf AVA-Seite
- ca. 6 Abnahmen / Mittwochsdoppelstunde

Qualität der Lösungen

ergeben Teilpunkte, Summe der Punkte eine Prozentzahl ➔ Teilnote

Bewertung der Abgabe

- 5 Punkte: Übung ist
  komplett, kommentiert, korrekt, skalierbar
- 4 Punkte:
  skaliert nicht oder kleinere Mängel

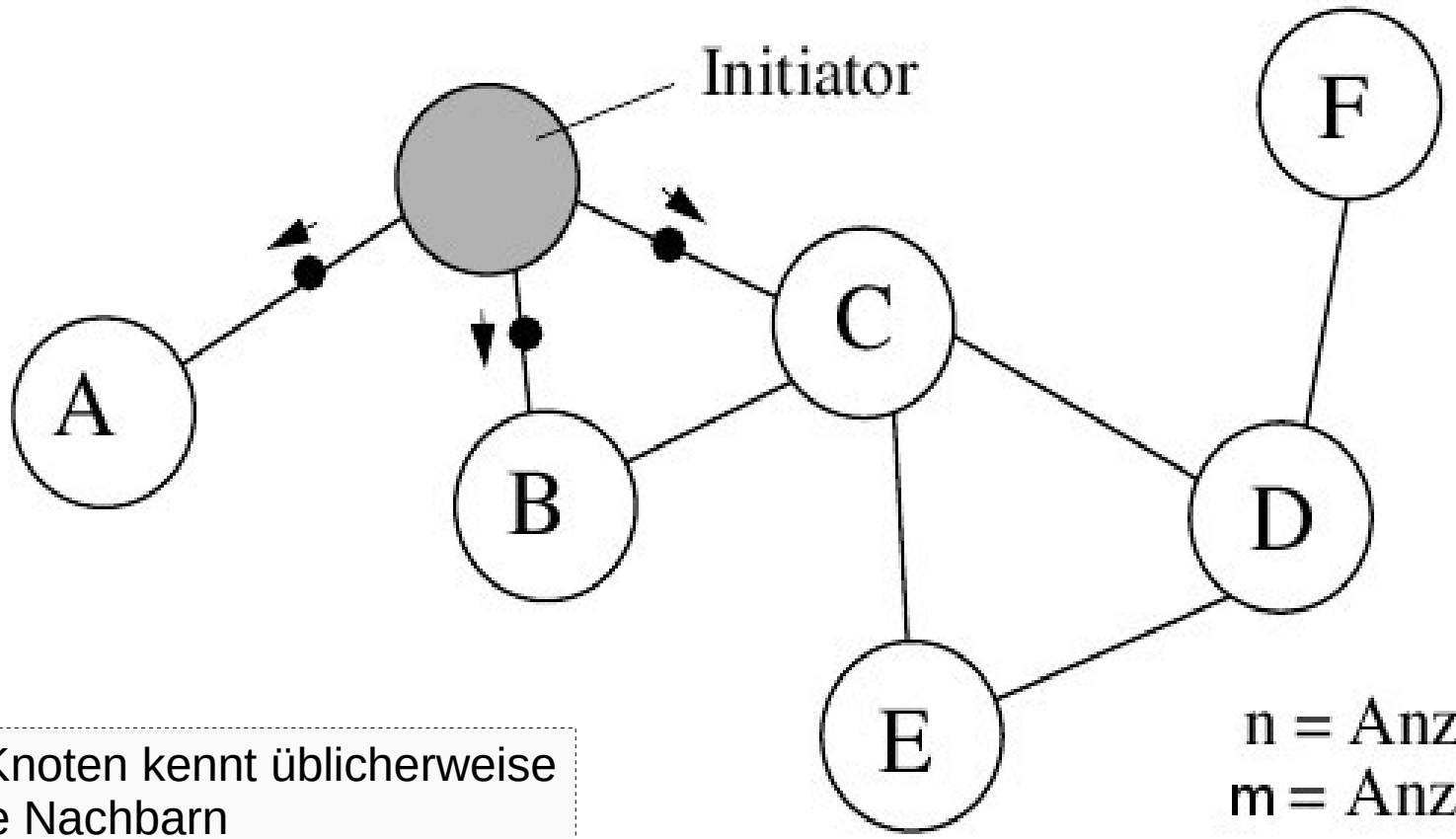
- 3 Punkte:
  korrekt für einen Spezialfall

- 2 Punkte:
  nicht besprochen, schwierig zu testen, Teile fehlen
- 1 Punkt, 0 Punkte: mangelhafte Umsetzung

Motivation

Einer hat die Info, jeder hat Freunde, Verbreitung eines Gerüchts.

Abstraktion



Ein Knoten kennt üblicherweise seine Nachbarn

n = Anzahl der Knoten
m = Anzahl der Kanten

Knoten = lokale Algorithmen Kanten=Verbindungen zum Nachrichtenaustausch

Algorithmus

```
R:  {Eine Nachricht <info> kommt an}
    if not informed then
       send <info> to all other neighbors;
       informed := true;
    fi
```

```
I:  {not informed}
    send <info> to all neighbors;
    informed := true;
```

Schwierigere Aufgabenstellung

Jeder habe eine Karte mit einer Zahl, gesucht ist die größte Zahl.

Noch schwieriger ...

Jeder hat Bargeld. Jeder schenkt jedem ständig Geld.

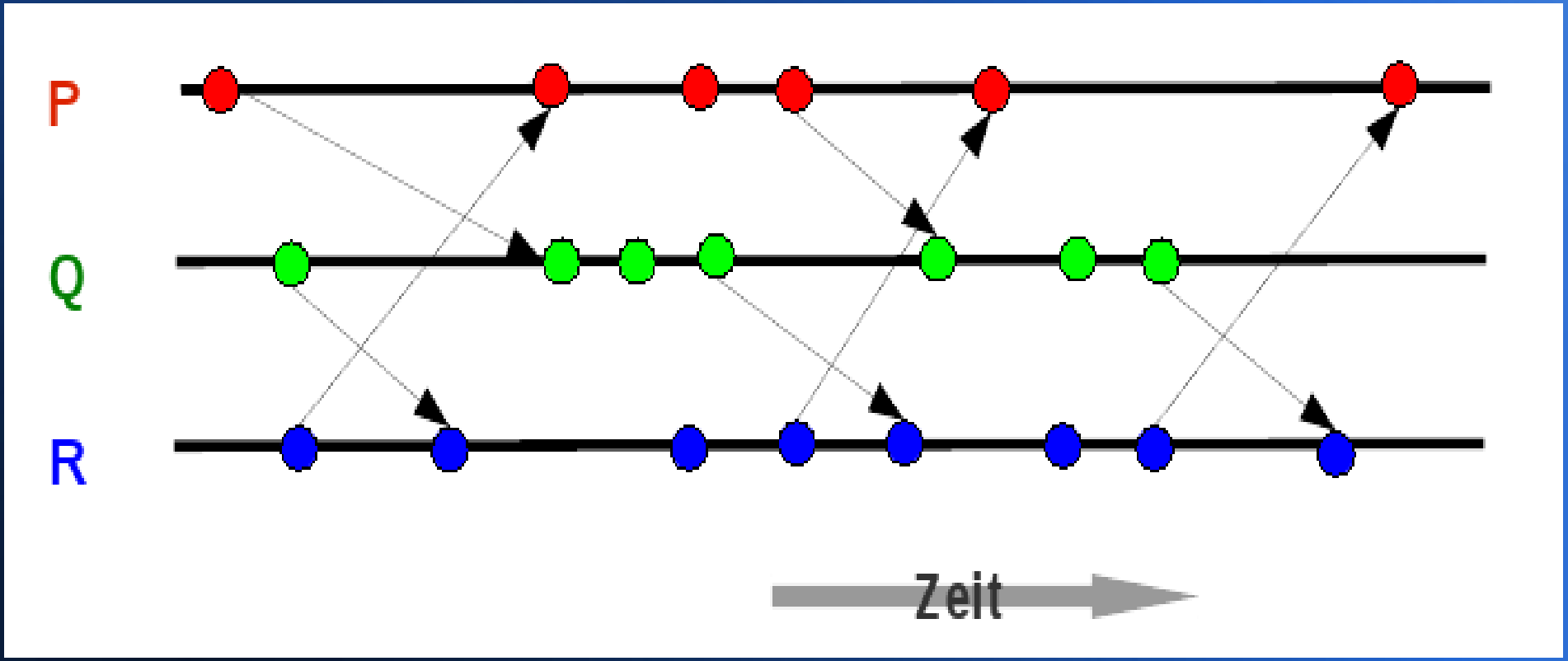
Wieviel Geld insgesamt im Umlauf?

Verteilte Algorithmen

- Algorithmen symmetrisch
  - gleichberechtigt
  - kein Client-/Server, sondern autonome Knoten
- Atommodell:
  - Rechnen in Nullzeit
  - Nachrichten mit endlicher Laufzeit
  - Nachrichten können sich nicht überholen (FIFO)

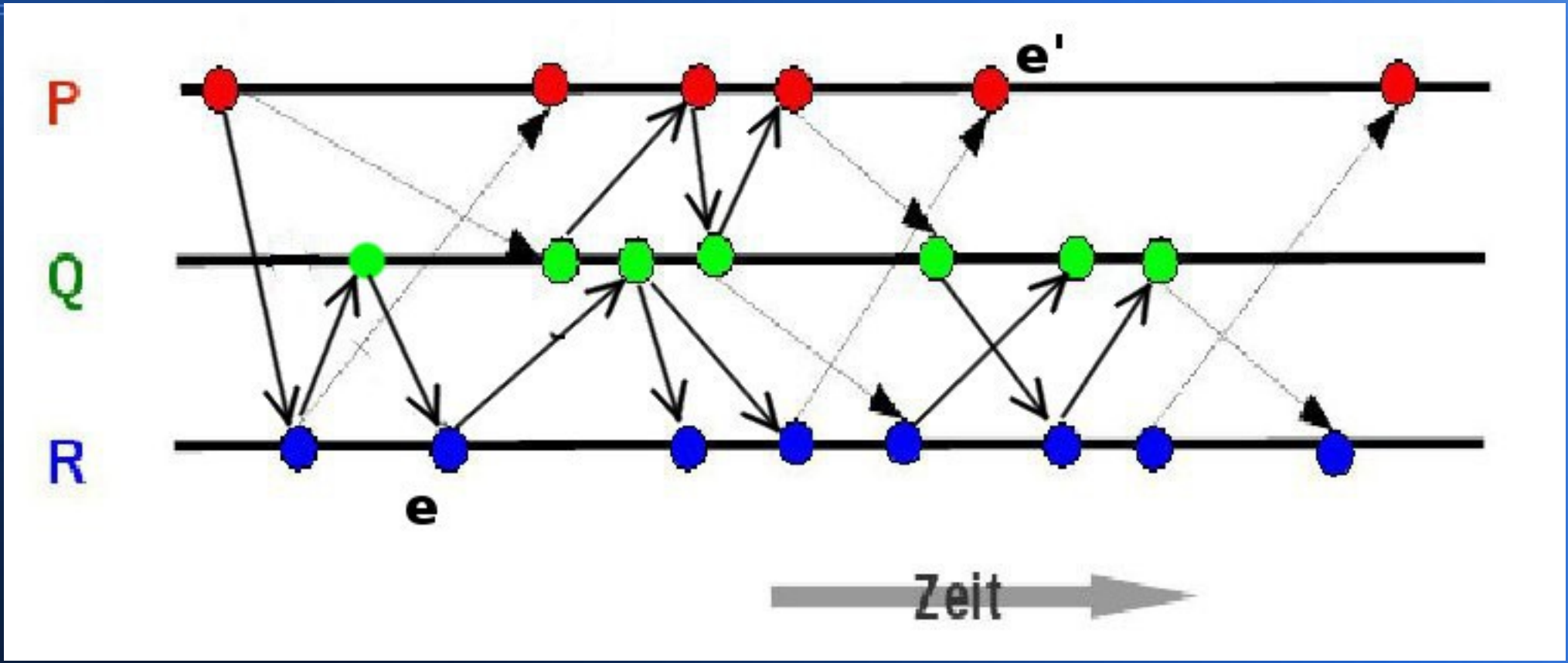
asynchrones Senden bzw. Empfangen

Raum-Zeit-Diagramme



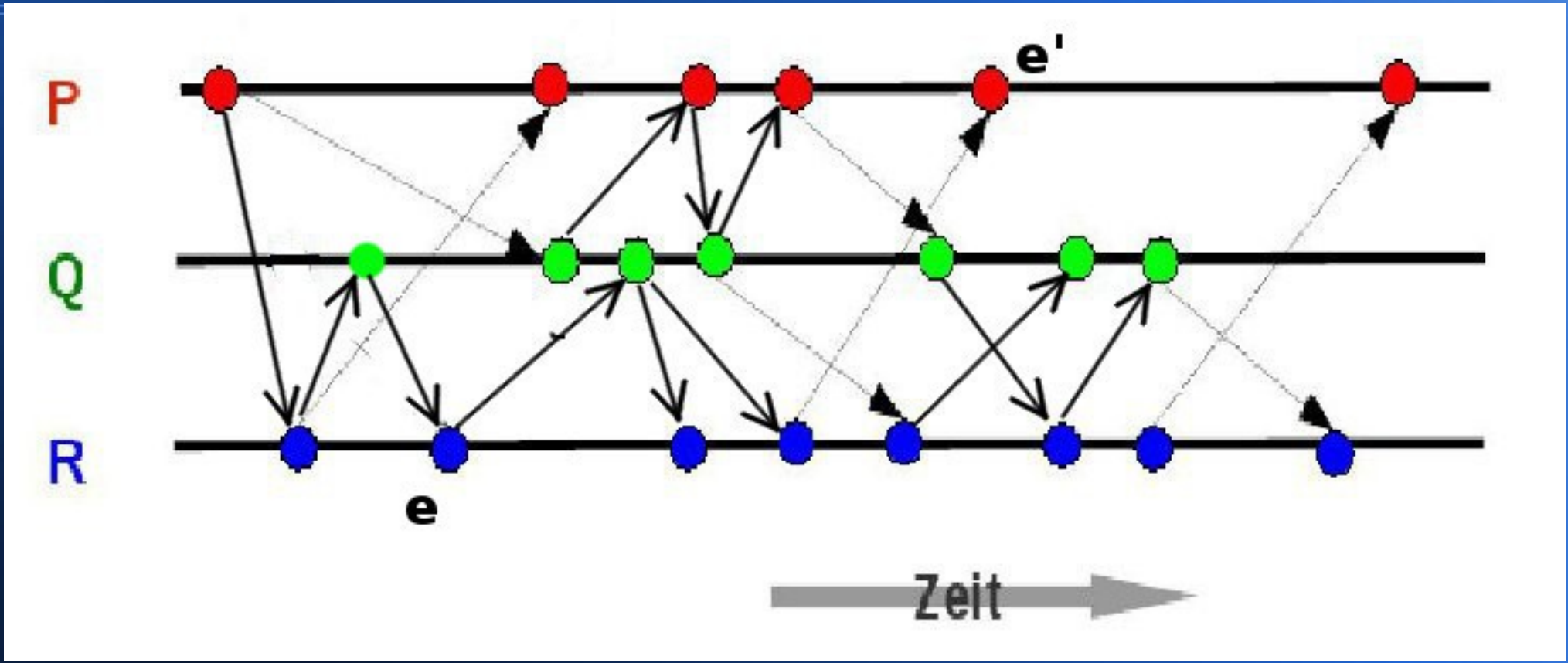
Knoten = Ereignisse, Kanten = Nachrichten mit Laufzeit (>0)

Kausalität



Ereignis e' ist kausal abhängig von e (e könnte e' beeinflussen)

Kausalität



Schreibweise: e' kausal abhängig von e ↔ e ≤ e'

Bedeutung: es gibt im Graphen einen Pfad von e zu e'

Kausalität erkennen?

- notwendig: „welches Ereignis ...“
  - war Ursache für einen Fehler ?
  - führte zur Überweisung von 6,7 Mio € ?
  - hat die Berechnung terminiert ?
  - ...
- ein Begriff von „das geschah vorher“ nötig
- Symbol „$\leq$“ beschreibt *happened-before-Relation*

Happened-Before

- zeitlich vorher, synchronisierte Zeit??
- zeitlich **und** logisch vorher
- kann auf zeitlich verzichten, weil verursachendes Ereignis sowieso vor dem resultierenden Ereignis liegt

Übungen: dringend notwendig

Übung 1

- Basisalgorithmus: autonomer Knoten
  - für das gesamte Semester benutzt
  - empfange Nachricht und sende an Nachbarn
- Erweiterung des Basisalgorithmus
  - graphviz
  - zufälligen Graphen erzeugen
  - Ausbreitung eines Gerüchts simulieren