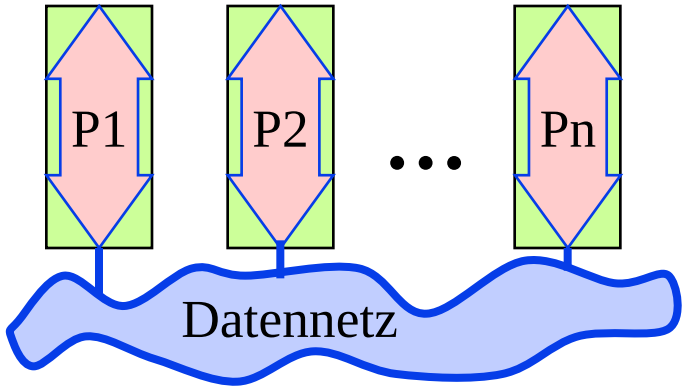
# Rechnernetze und verteilte Systeme (BSRvS II)

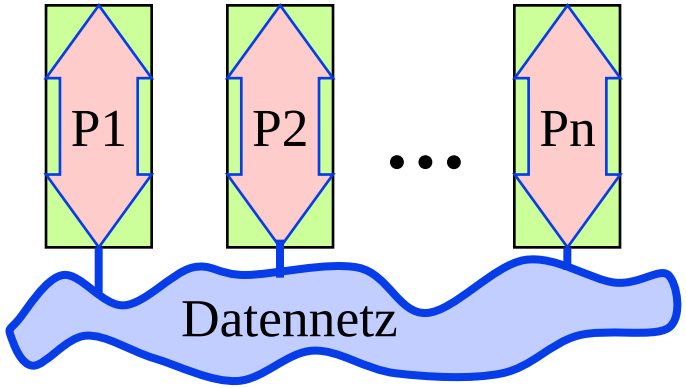



- Allgemein
- Atommodell
- Echo
- Vertreterauswahl
  - Maximumsuche
- Schnappschuss
  - Kausale Abhängigkeit

**Prof. Dr. Heiko Krumm**
**FB Informatik, LS IV, AG RvS**
**Universität Dortmund**

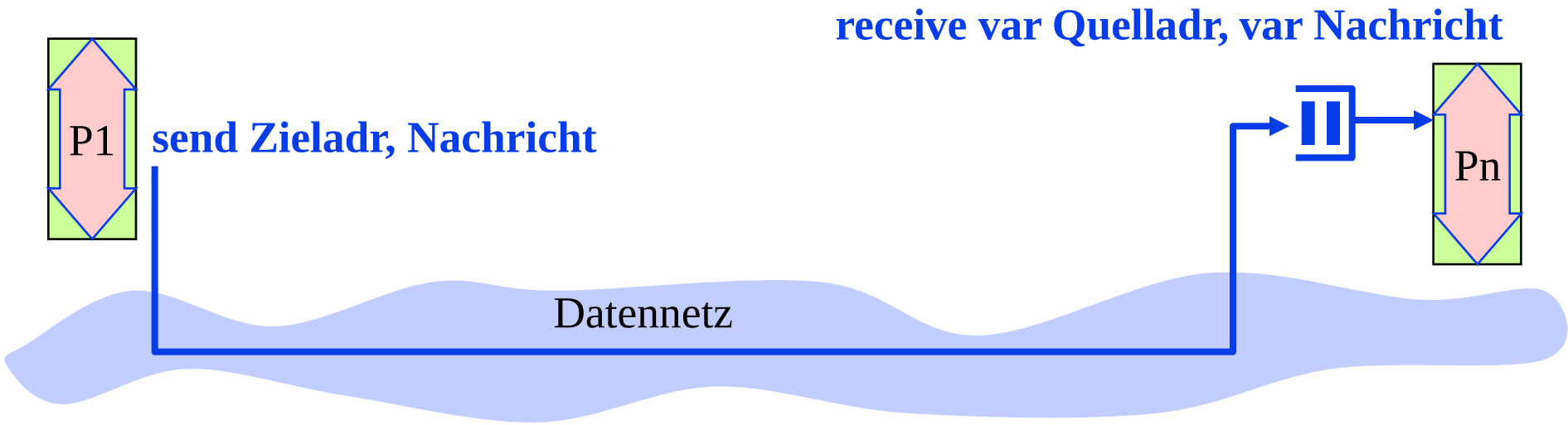
- **Computernetze und das Internet**
- **Anwendung**
- **Transport**
- **Vermittlung**
- **Verbindung**
- **Multimedia**
- **Sicherheit**
- **Netzmanagement**
- **Middleware**
- **Verteilte Algorithmen**

# Verteilter Algorithmus




- Abläufe in den Anwendungsprozessen einer verteilten Anwendung unter Abstraktion von Implementierungsdetails
  - Je Station: 1 Knotenprozess der Knotenalgorithmus ausführt
    Nur lokale (private Variablen), sequentielle Ausführung je Station
  - Knotenprozesse interagieren: Sie tauschen Nachrichten aus
- Verteilter Algorithmus ::=
  Menge von Knotenalgorithmen
  Nachrichtenaustausch zwischen Knotenalgorithmen

# Verteilter Algorithmus: Kommunikation

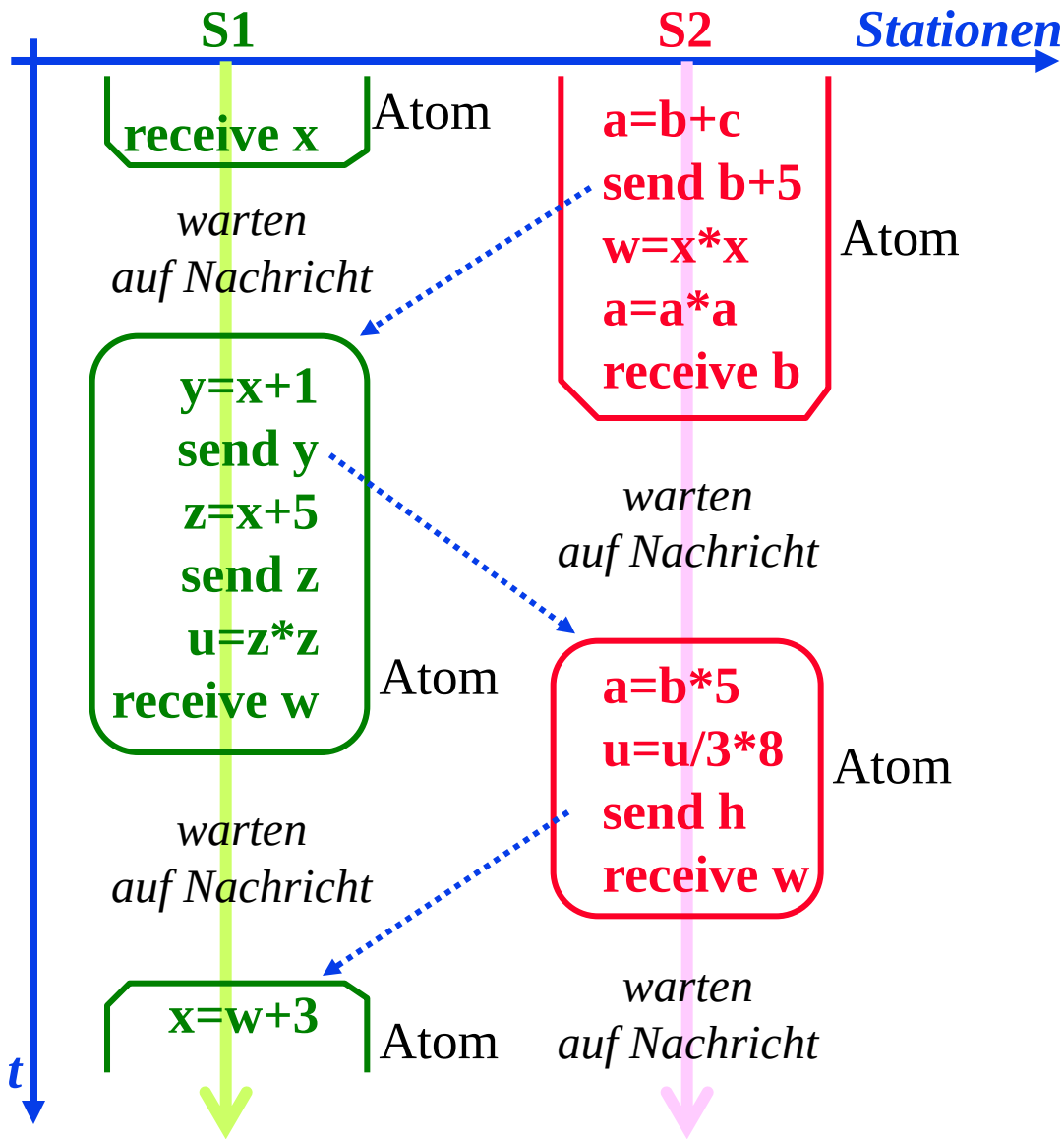



**Datagramm-Austausch**

- nicht reihenfolgetreu, aber oft Annahme, dass sich Nachrichten zwischen denselben Endpunkten nicht überholen
- Laufzeit nicht vorhersagbar
- Varianten
  a) zuverlässig
  b) Verluste möglich

# Verteilter Algorithmus: Atommodell

- Abstraktion von lokalen Ausführungszeiten
  - Zeit vergeht beim Transfer von Nachrichten und beim Warten auf Nachrichten
  - Lokale Aktionsfolgen werden zeitlich punktuell durchgeführt

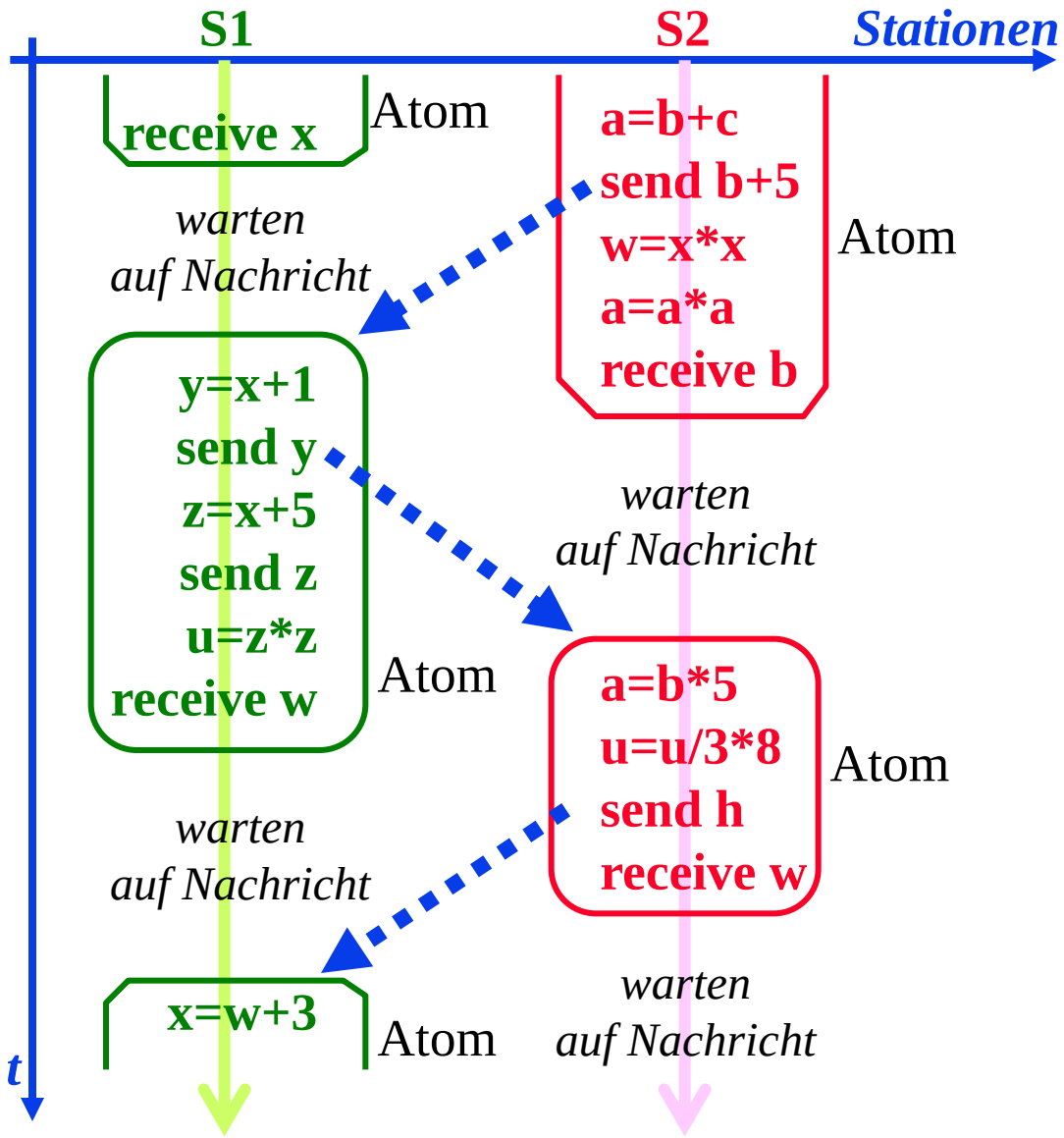


# Verteilter Algorithmus: Atommodell

**5 zeitlich punktuelle Ereignisse**

- Jedes Ereignis besteht aus dem Empfang und der lokalen Verarbeitung einer Nachricht

- Die Pfeile „**A** ▪▪▪▶ **B**“ verdeutlichen Kausalitätsbeziehung zwischen Ereignissen, d.h. **A** sendet Nachricht, die mit **B** empfangen wird



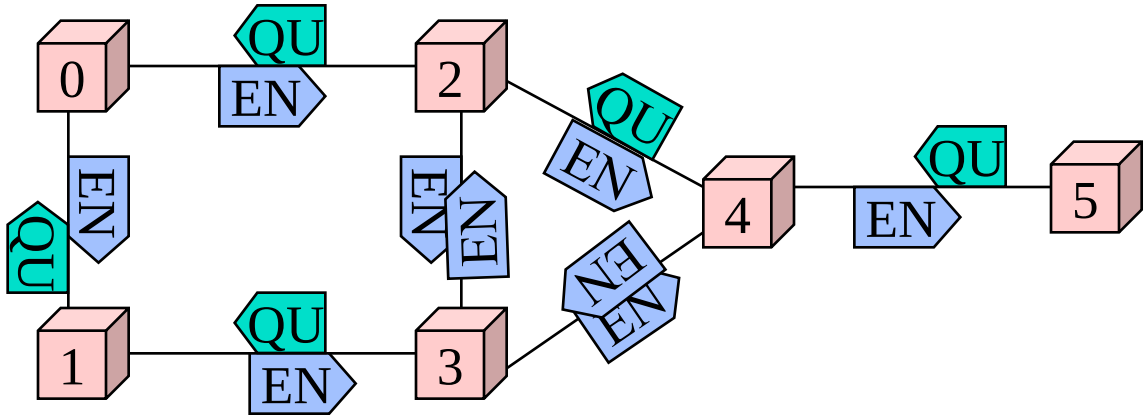
# Verteilter Algorithmus: Aufwand / Komplexität

- Bei sequentiellen Algorithmen:
  Zeit und Speicherbedarf

- Bei verteilten Algorithmen:
  Kommunikation ist teuer
  - Anzahl der Nachrichten
  - Länge der Nachrichten
  - Zeit für den Nachrichtenaustausch

# Beispiel: Echo-Algorithmus

- Ermitteln eines (nachrichtenlaufzeitoptimalen) Spannbaums
- Durchführen eines bestätigten Broadcasts




- Algorithmus
  - Initiator sendet Erkundungsnachricht EN an alle seine Nachbarn.
  - Jede Station, die EN erstmals empfängt, merkt sich den Absender M und sendet EN an alle übrigen Nachbarn.
  - Stationen ohne übrige Nachbarn senden sofort Quittung QU an M zurück.
  - Wenn Station von jedem übrigen Nachbarn QU oder EN empfing, sendet sie QU an M.
  - Initiator schließt erfolgreich ab, wenn von allen seinen Nachbarn EN oder QU eingetroffen.
  - Die Wege der QUs bilden den Spannbaum.

# Echo-Algorithmus

- siehe auch: http://ls4-www.informatik.uni-dortmund.de/RVS/MA/hk/OrdnerVertAlgo/VertAlgo.html
- Es seien n die Anzahl der Knoten und e die Anzahl der Kanten im Netz
- Nachrichtenanzahl
  - Es werden auf jeder Kante genau zwei Nachrichten ausgetauscht: Anzahl Nachrichten = 2 * e
- Nachrichtenlänge
  - Die Erkundungsnachrichten EN haben alle dieselbe konstante Länge
  - Annahme: Mit den Quittungsnachrichten wird die Spannbaum-Information zurückgespielt, z.B. als strukturierte Knotenliste ➔ max. Länge einer QU-Nachricht wächst mit n
- Dauer in Anzahl von seriellen Nachrichtenaustausch-Vorgängen
  - Im schlimmsten Fall ist das Netz zur Liste entartet ➔ max. Ablaufdauer < 2*n zeitlich hintereinander liegende Austauschvorgänge

# Vertreter-Auswahl

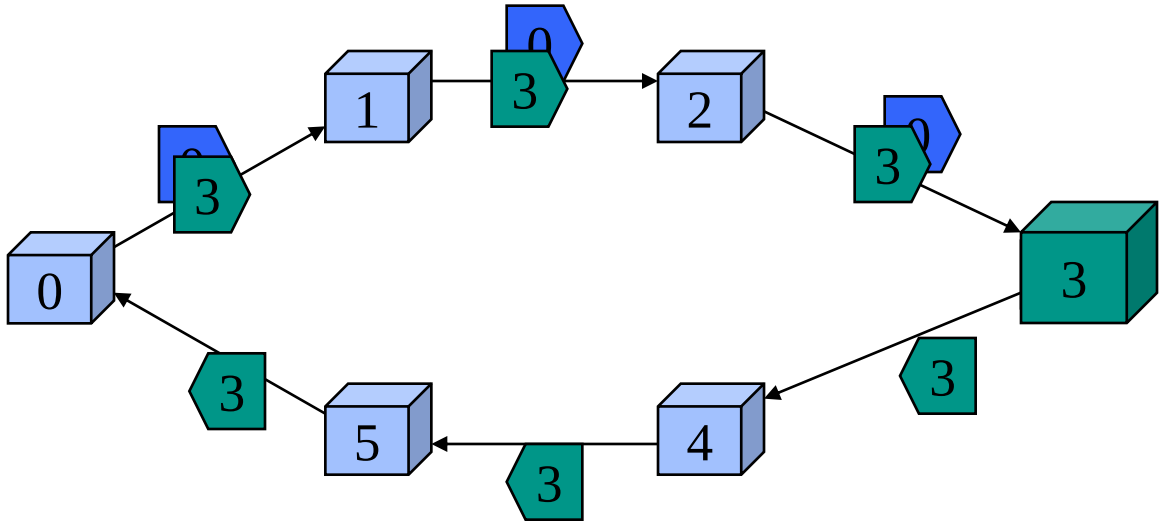
- Schritt zur Zentralisierung einer Problemlösung
  Bestimme genau einen Repräsentanten unter den Knoten
- gegeben: Menge gleichberechtigter Stationen
- gesucht:
  Genau 1 Station als Vertreter, auch wenn gleichzeitig mehrere Stationen die Vertreterauswahl aktivieren
  - Es gibt gerade 1 Initiator: Wähle diesen als Vertreter
  - Es gibt mehrere nebenläufige Initiatoren: Wähle einen eindeutig aus.
- Lösungsidee
  - geordnete Attributmenge mit eindeutiger Zuordnung zu Stationen
  - Maximumsuche
    - » Beispiele für Attributmengen
      Stationsnetzadresse
      Verbund aus Problemgröße (z.B. Freispeichervorrat) und Stationsadresse

# Maximum-Suche

- gegeben: Menge gleichberechtigter Stationen, eindeutige Zuordnung von Attributwerten zu Stationen, totale Ordnung der Attributwerte
- gesucht:
  Finde maximalen Attributwert und mache ihn allen bekannt
- Lösungsidee
  - Jede Station speichert initial den Wert -∞ als bisher bekanntes Maximum
  - Initiator sendet seinen (echten) Wert ($\neq -\infty$) an alle seine Nachbarn
  - Station, die einen Wert empfängt, bildet das Maximum aus dem empfangenen und ihrem bisher gespeicherten Maximum. Sie aktualisiert gegebenenfalls ihr gespeichertes Maximum und sendet dann auch (aber nur dann) das neue ihr bekannte Maximum an alle ihre Nachbarn (außer an den, von welchem sie das neue Maximum empfing).
  - Irgendwann ist bei allen Stationen das tatsächliche Maximum hinterlegt.
  - Problem: Wer erkennt, wann das eingetreten ist.
    Ein neues Problem: Verteilte Terminierungserkennung

# Maximum-Suche

- Probleme
  1. Terminierungserkennung
  2. Nachrichtenaufwand (nebenläufige Initiatoren)
- Lösungsmöglichkeit: Spezielle Netztopologie
  **Maximumsuche im unidirektionalen Ring**

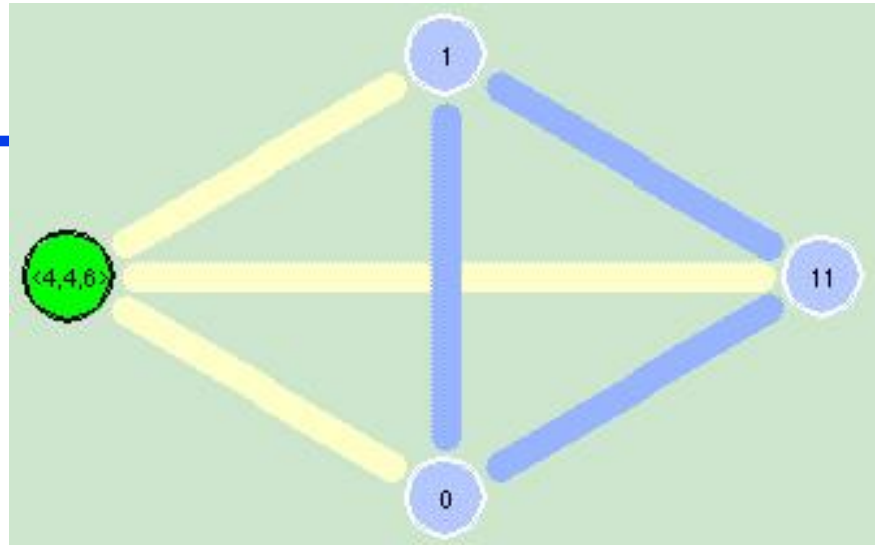



  - Initiator, der eigenen Wert empfängt, erkennt Terminierung
  - Weitergabe nur jeweils an 1 Nachbarn

# Verteilter Schnappschuss

- Eine Station eines verteilten Systems möchte eine Kopie des globalen Zustands, um ihn lokal auswerten zu können: **Schnappschuss**
- Schnappschuss ::=
  Vektor aus Stationszustandskopien
- Eigenschaften eines Schnappschuss
  - aktuell oder veraltet
  - gleichzeitig oder nicht:
    Wurden alle lokalen Kopien zum selben Zeitpunkt erzeugt?
- Gleichzeitiger Schnappschuss ist teuer: Anhalten von Stationen
- Nicht-Gleichzeitiger Schnappschuss kann Falsches widerspiegeln
  - Beispiel:
    Volkszählung, dadurch, dass Zähl-Beamte die Wohnungen aufsuchen und die Anwesenden abzählen.
    Wenn keine Ausgangssperre verhängt wird, können Personen mehrfach gezählt werden.

# Verteilter Schnappschuss

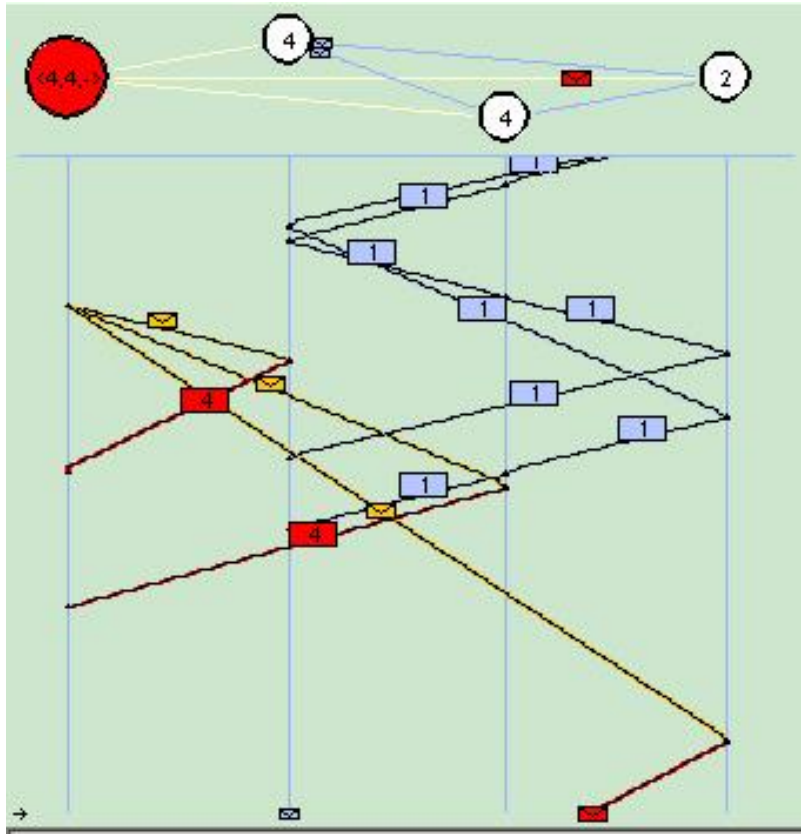



- Schnappschuss-Konsistenz
  - Verteilter, u.U. nicht-gleichzeitiger Schnappschuss soll „Nichts Falsches widerspiegeln“
  - Die Konsistenz gewährleistet nicht, dass der durch den Schnappschuss dargestellte Globalzustand beim tatsächlichen Ablauf des Systems auch eingenommen wurde.
    Sie stellt aber sicher, dass der Zustand hätte auftreten können, wenn die Nachrichtenlaufzeiten im System geeignet variiert worden wären.
  - Der von einem konsistenten Schnappschuss dargestellte Globalzustand liegt im Spielraum, den der verteilten Systemen innewohnende Nichtdeterminismus offenlässt.
- Beispiel (oben) für Inkonsistenz:
  Quersumme des Schnappschusses ist 14, es sind aber nur 12 im System, d.h. 2 wurden doppelt gezählt.
  - siehe http://ls4-www.informatik.uni-dortmund.de/RVS/MA/hk/OrdnerVertAlgo/VertAlgo.html

# Verteilter Schnappschuss: Einfrieren

- Aktueller und gleichzeitiger Schnappschuss durch Einfrieren
  - Initiator sendet Anforderung RQ an alle anderen Stationen und wartet dann, bis von allen Stationen bei ihm eine Antwort RP eingetroffen ist.
  - Station, die RQ empfängt, stellt jede Aktivität ein. Sie erzeugt lediglich eine Zustandskopie und sendet sie per Reply-Nachricht RP an den Initiator. Danach wartet sie auf eine Freigabe-Nachricht AC.
  - Initiator sendet, nachdem er alle RPs empfing, an alle Stationen eine AC-Nachricht.

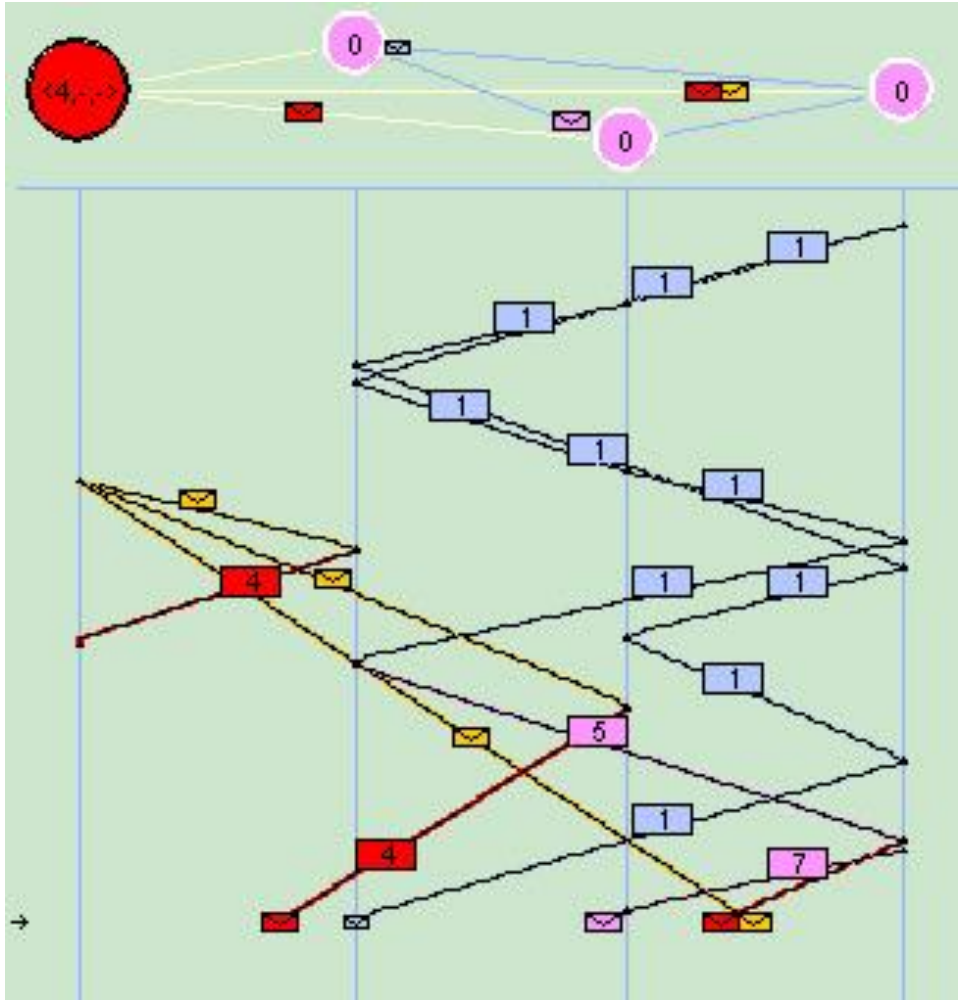
➔ Es gibt einen Zeitpunkt, an welchem alle Stationen warteten und den in den RP-Nachrichten gemeldeten Zustand hatten



# Verteilter Schnappschuss: Weitergabe

- Konsistenter Schnappschuss durch Weitergabe
  - Initiator sendet Anforderung RQ an alle anderen Stationen und wartet dann, bis von allen Stationen bei ihm eine Antwort RP eingetroffen ist.
  - Station, die RQ empfängt, erzeugt RP sofort und merkt sich, dass Schnappschuss aktiv ist: Jede Nachricht, die sie versendet, versieht sie mit einer entspr. Kennung.
  - Station, die Nachricht mit Kennung empfängt, verfährt so, als ob sie vor Empfang dieser Nachricht ein RQ bekommen hätte.

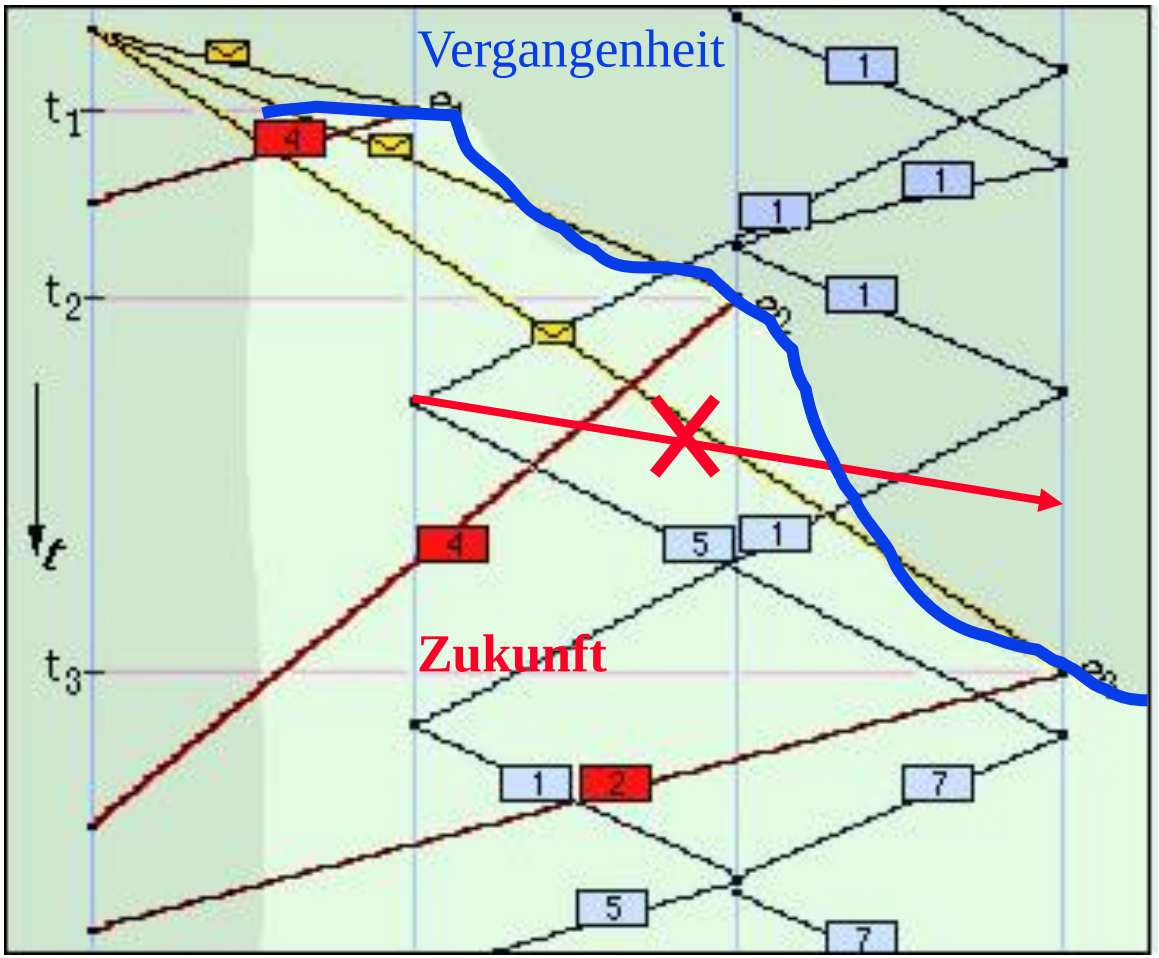
➔ „Besucher wird nicht doppelt gezählt!“



# Schnappschuss: Konsistenz – informell

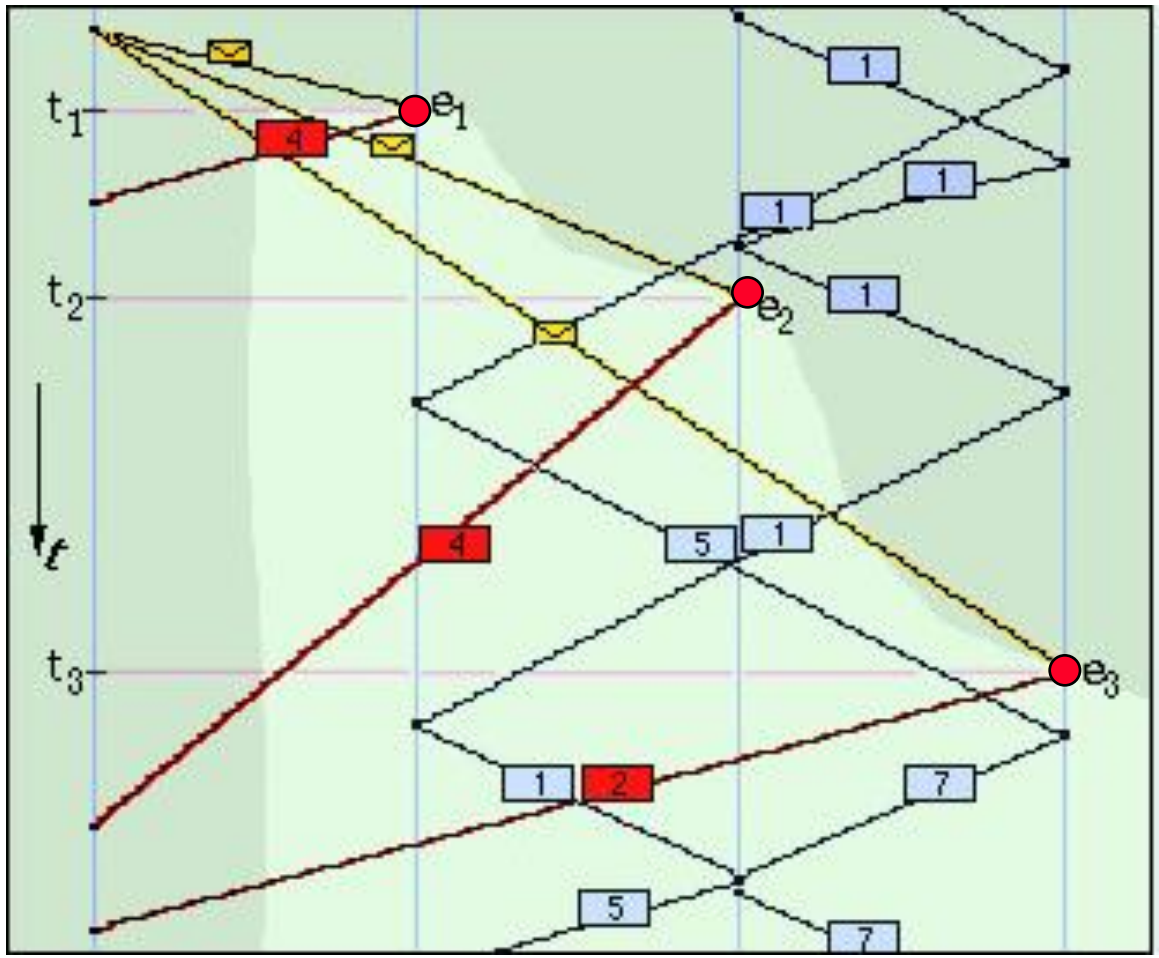
- Vergangenheit / Zukunft eines Schnappschusses
  - dunkler / heller Bereich im Weg/Zeit-Diagramm

- Schnappschuss ist konsistent ⇔
  **Es gibt keine Nachrichten, die in der Zukunft gesendet und in der Vergangenheit empfangen werden.**



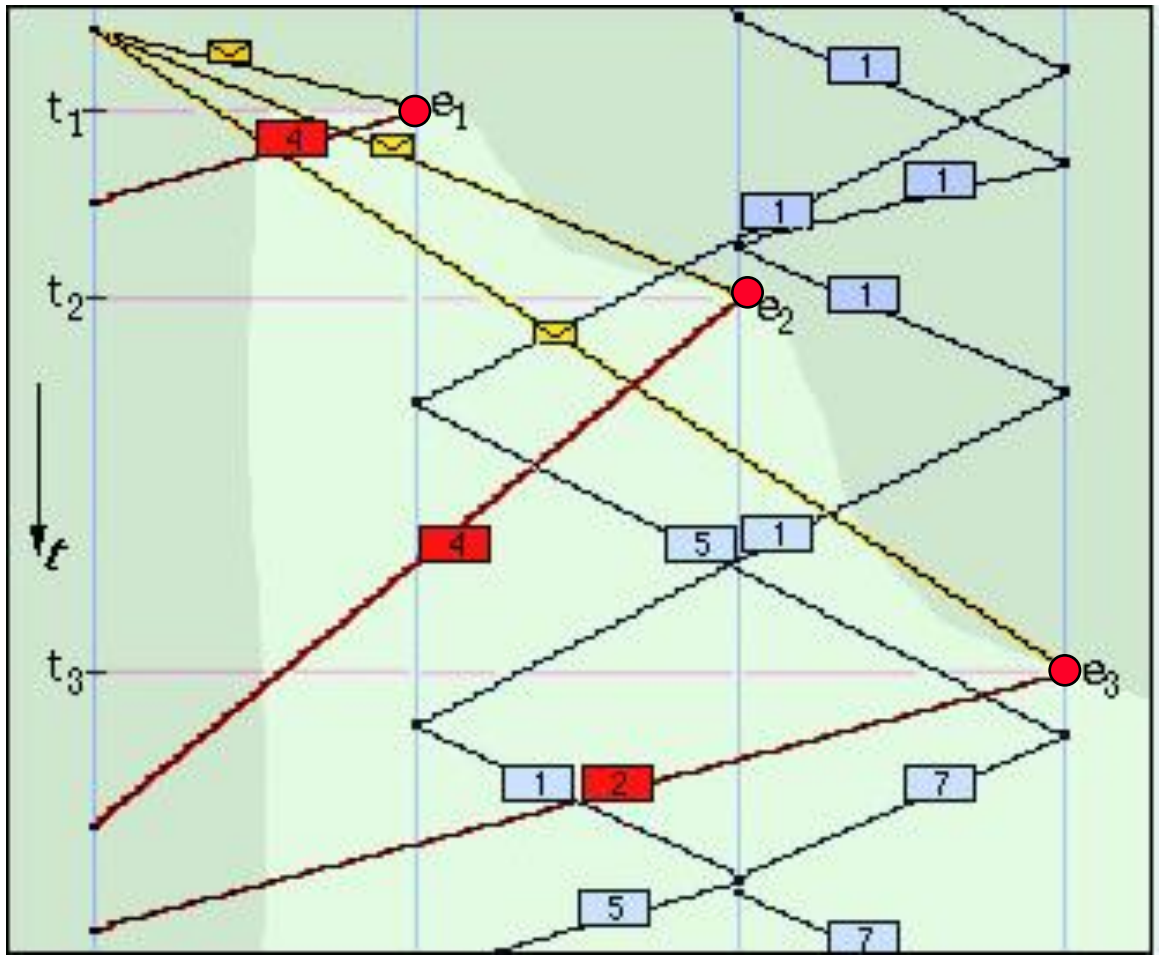
# Schnappschuss: Konsistenz – formal

- Ereignisse, mit denen die lokalen Kopien erzeugt werden:
  - $e_1, e_2, e_3$
- Zeitpunkte dieser Ereignisse
  - $t_1, t_2, t_3$
- Schnappschuss ist konsistent ⇔
  Es muss im nichtdeterministischen Spielraum des Systems liegen, dass die $\mathbf{e_i}$ alle gleichzeitig sind ($t_i=t_j$)



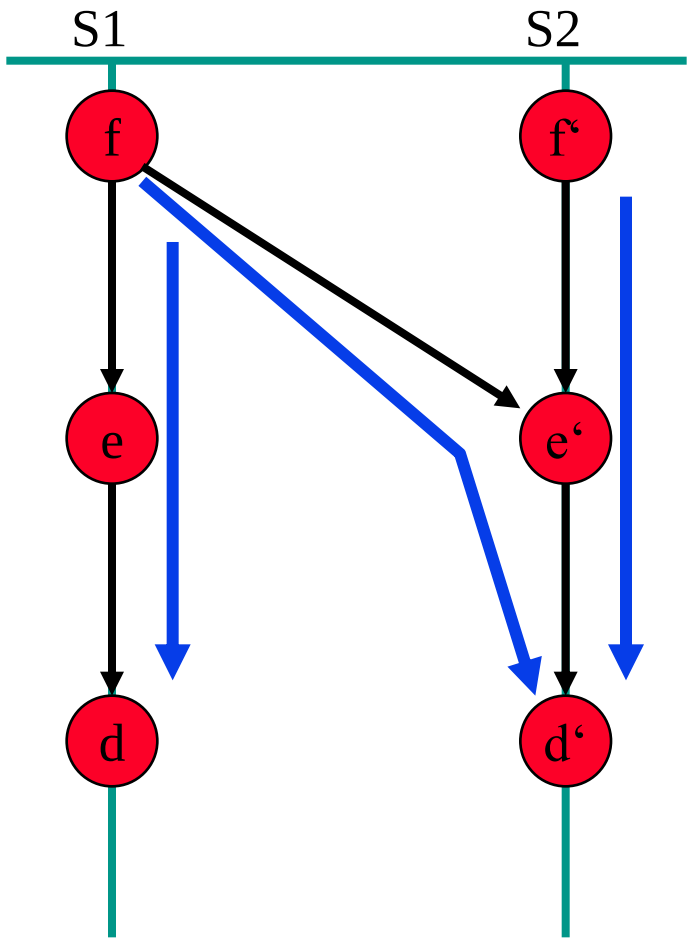
# Schnappschuss: Konsistenz – formal

- Ereignisse, mit denen die lokalen Kopien erzeugt werden:
  - $e_1$, $e_2$, $e_3$
- Zeitpunkte dieser Ereignisse
  - $t_1$, $t_2$, $t_3$
- Schnappschuss ist konsistent ⇔
  Alle $e_i$ sind zueinander paarweise **kausal unabhängig**.



# Kausale Abhängigkeit von Ereignissen im verteilten System

- Ereignis e ist direkt kausal abhängig von Ereignis f, wenn
  - e und f in derselben Station stattfinden und f dem e unmittelbar vorausgeht (lokale direkte Abhäbigkeit)
  - e eine Nachricht empfängt, die mit dem Ereignis f gesendet wurde
- Ereignis e ist von f kausal abhängig:
  - Transitive Hülle der direkten Abhängigkeit

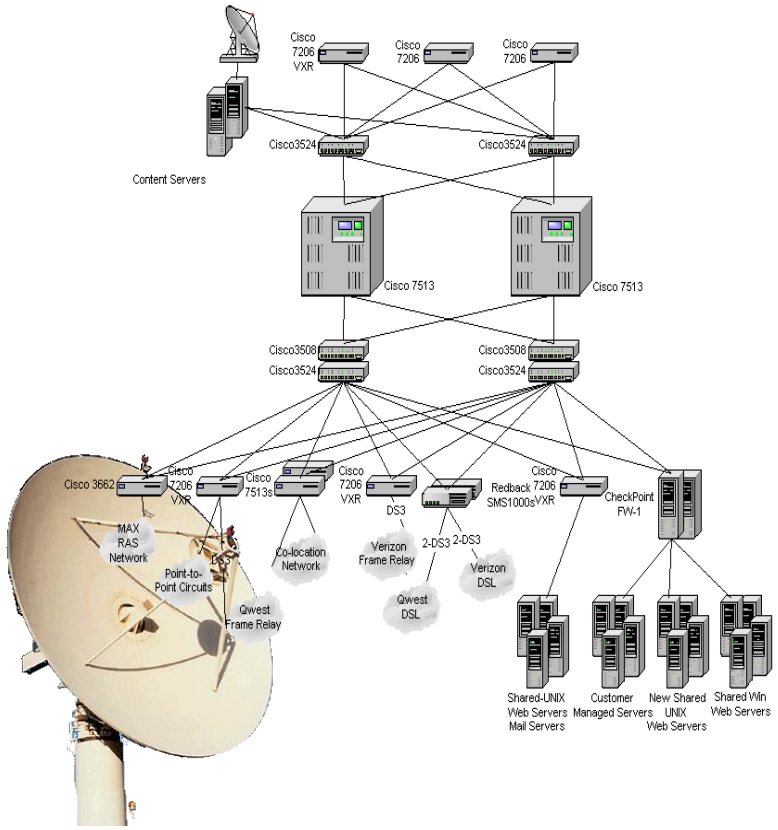



***e von f kausal abhängig:***
***f kann e beeinflussen***

# Verteilte Algorithmen

- Das war ein kurzer Einblick
- Es gibt noch viel mehr, z.B.:
  - Terminierung
  - Synchronisationsbedingungen
  - Gegenseitiger Ausschluss
  - Verklemmungen
  - Konsensus
  - Verteilte Transaktionen
  - Netzerkennung
  - Routing (schon bekannt)
  - …

# Betriebssysteme, Rechnernetze und verteilte Systeme II (BSRvS II)




**Prof. Dr. Heiko Krumm**

**FB Informatik, LS IV, AG RvS**
**Universität Dortmund**

- **Computernetze und das Internet**
- **Anwendung**
- **Transport**
- **Vermittlung**
- **Verbindung**
- **Multimedia**
- **Sicherheit**
- **Netzmanagement**
- **Middleware**
- **Verteilte Algorithmen**