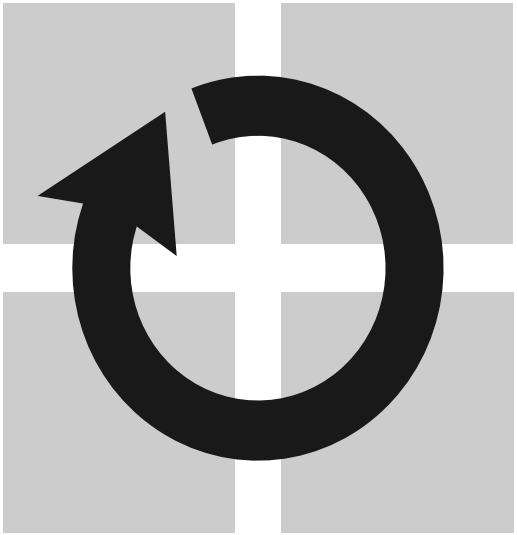
Fehlertolerante verteilte Systeme, Peer-To-Peer Netzwerke

Hauptseminar im SS 2002

Hans Reiser, Rüdiger Kapitza



Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme
Universität Erlangen-Nürnberg

Überblick - Teil 1

- Zuverlässigkeit und das World Wide Web
- Grundlegendes zur Fehlertoleranz
- Fehler- und Systemmodelle

Zuverlässigkeit und das World Wide Web

- World Wide Web als Freizeitbeschäftigung
  - Im Vordergrund: Bunt, billig, schnell, komfortabel zu verwenden
  - Kurzzeitige Ausfälle haben keine grosse Auswirkung
- World Wide Web als Geschäftsbasis für verteilte Anwendungen
  - Gewinnt immer mehr an Bedeutung
  - Zuverlässigkeit/Verfügbarkeit wird zur zentralen Anforderung

Zuverlässigkeit und das World Wide Web

- Beispiel Online Banking
  - Zeitkritische Börsengeschäfte
  - Attraktiv für kriminelle Machenschaften
- Beispiel Medizin
  - Hohe Ansprüche an Genauigkeit und Verfügbarkeit von Daten
  - Strenge rechtliche Vorschriften an Sicherheit und Datenschutz
- Beispiel Online-Shops
  - Direkte finanzielle Folgen von Ausfällen

Verlässliche verteilte Systeme nicht einfach

- Neuentwicklung eines Air Traffic Control-Systems der USA ab 1989:
  - Verwendung von Standard-Hardware (einfache Wartung, Upgrade, Verwaltung)
  - Web-ähnliche GUI
- IBM wollte Software für Fehlertoleranz und Konsistenz erstellen
- Projekt wurde grosses Fiasko
  - IBM scheiterte mit der Entwicklung, die Probleme waren viel grösser als erwartet

Heutige Situation

- Sicherheit wird zumindest von manchen Softwareherstellern als wichtig betrachtet
- Zuverlässigkeit ist in der Softwareentwicklung nur selten ein beachtetes Thema

Grundlegendes zur Fehlertoleranz

- Leslie Lamport:

  A distributed system is one in which the failure of a machine you have never heard of can cause your machine to become unusable.

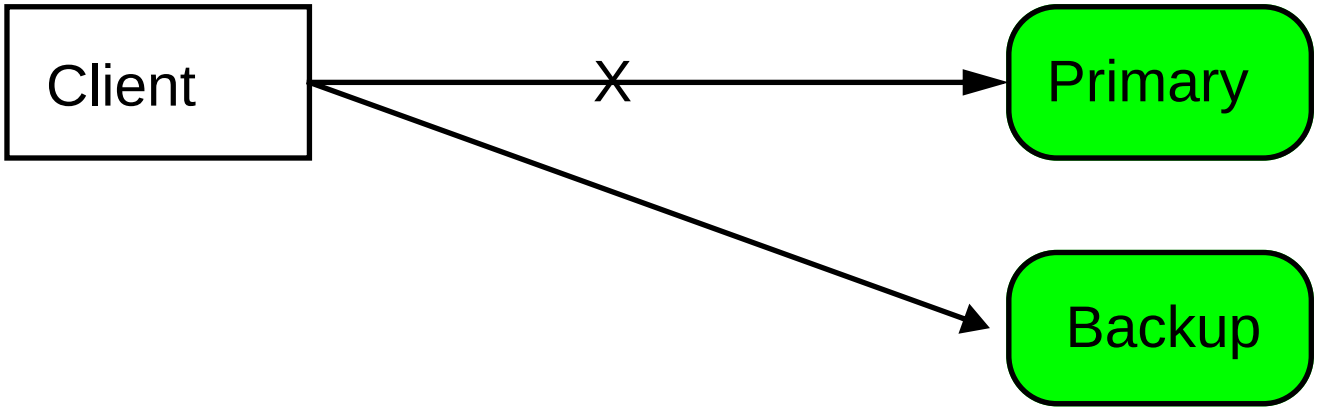
- anders ausgedrückt:
  - Ein zentrales System ist nicht fehlertolerant, da es einen “single point of failure” besitzt.
  - Ein verteiltes System, ohne geeignete Massnahmen, besitzt viele “single point of failure”.
  - Zuverlässigkeit und Fehlertoleranz sind also keine impliziten Eigenschaften von verteilten Systemen, sondern müssen erst hart erkämpft werden.

Grundlegendes zur Fehlertoleranz

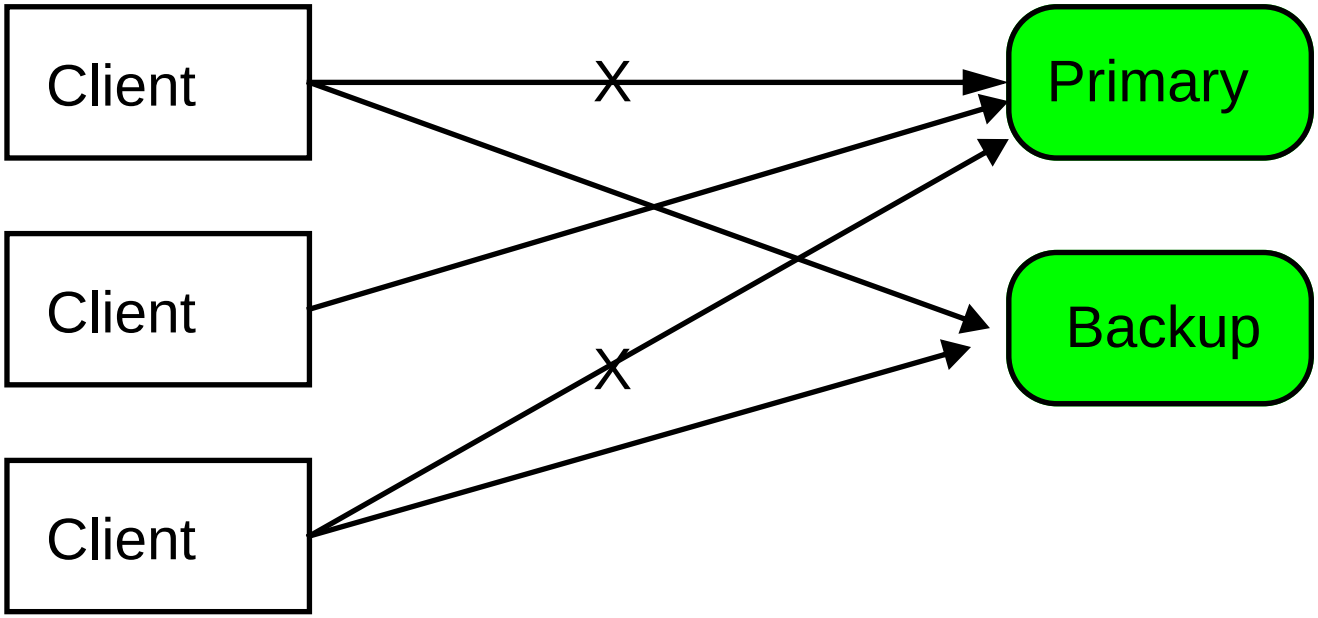
- Grundprinzipien
  - Dezentralisierung: Bei Ausfällen nur Teile des Systems betroffen
  - Redundanz/Replikation zur Maskierung von Ausfällen

Einfaches Primary-Backup-Modell

Einfaches Primary-Backup-Modell



Einfaches Primary-Backup-Modell



... bei mehreren Klienten entstehen schnell Inkonsistenzen

“Fehlertolerantens” 2-Phase-Commit

- Zufällig gewählter Knoten übernimmt Koordination einer Operation
- Prepare muss von Mehrheit der Knoten akzeptiert werden
  - keine Inkonsistenz durch unterschiedliche Koordinatoren
- Bei Akzeptanz: Commit durch den Koordinator

“Fehlertolerantens” 2-Phase-Commit

- Zufällig gewählter Knoten übernimmt Koordination einer Operation
- Prepare muss von Mehrheit der Knoten akzeptiert werden
  - keine Inkonsistenz durch unterschiedliche Koordinatoren
- Bei Akzeptanz: Commit durch den Koordinator
- Single Point of Failure:
  - Bei Ausfall des Koordinators zwischen Prepare und Commit: System blockiert sich

Modellierung: Fehlermodelle

- “Fail-Stop”-Fehlermodell
  - Rechner bleiben stehen, sobald irgend ein Problem auftritt
  - Dinge wie Checksummen, Parity, etc. können helfen, diesem Ideal nahe zu kommen
  - ansonsten: Hoffen, dass sich die Realität an das Modell hält?

Modellierung: Fehlermodelle

- In der Realität
  - Hardware-Fehler werden nicht, oder erst zu spät erkannt (wenn der Zustand bereits korrumpiert ist)
  - Software-Fehler gibts es in jedem grösserem Projekt, und zeigen sich selten als “Fail-Stop”-Verhalten
  - Hacker-Angriffe versuchen gezielt, das System zu manipulieren, ohne dass dies bemerkt wird.
- Byzantinisches Fehlermodell
  - beliebiges Fehlverhalten wird erwartet (allgemeinster Fall!)
  - wesentlich schwieriger zu behandeln

Modellierung: Systemmodelle

- Synchrones Modell
  - Garantierte zeitliche Aussagen zum Systemverhalten (z.B. Nachrichtenlaufzeit, Uhrensynchronisation zwischen den Rechnern)
- Asynchrones Modell
  - Rechenzeit und Kommunikation unterliegt keinen deterministischen Einschränken
  - Jedoch: Theoretische Machbarkeitsgrenzen (siehe Vortrag am 29. April)

Ziele des Seminars

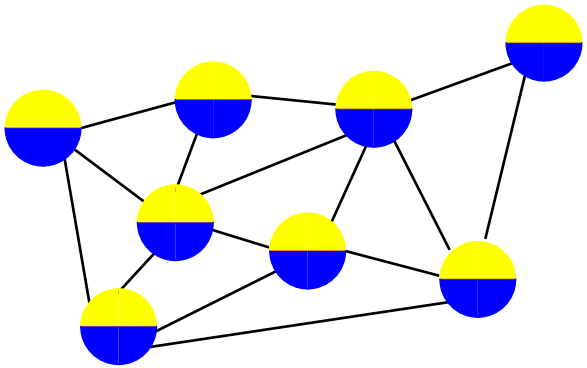
- Grundlagen der Fehlertoleranz in verteilten Systemen
  - MAFTIA-Systemmodell, Theoretische Machbarkeitsgrenzen
- Aktuelle Forschung im Bereich praktikable Algorithmen für fehlertolerante Internet-Anwendungen, Tolerierung von “bösartigem” Fehlverhalten (byzantinische Fehler)
  - PAXOS, Practical Byzantine Fault Tolerance, Totem&SecureRing, Cachin/Shoup/Kursawe
- Peer-To-Peer-Netzwerke als aktueller Trend in Richtung völliger Dezentralisierung und als Basistechnologie für zukünftige verteilte, hochverfügbare Systeme
- Gnutella, JXTA&Poblano, Freehaven&Publius, Freenet, Pastry&Scribe, Chord&CFS

Überblick - Teil 2

- Motivation (oder Gründe für den Hype)
- Definition des Begriffs Peer-to-Peer
- Anwendungen und Anforderungen
- Ziele des Seminars

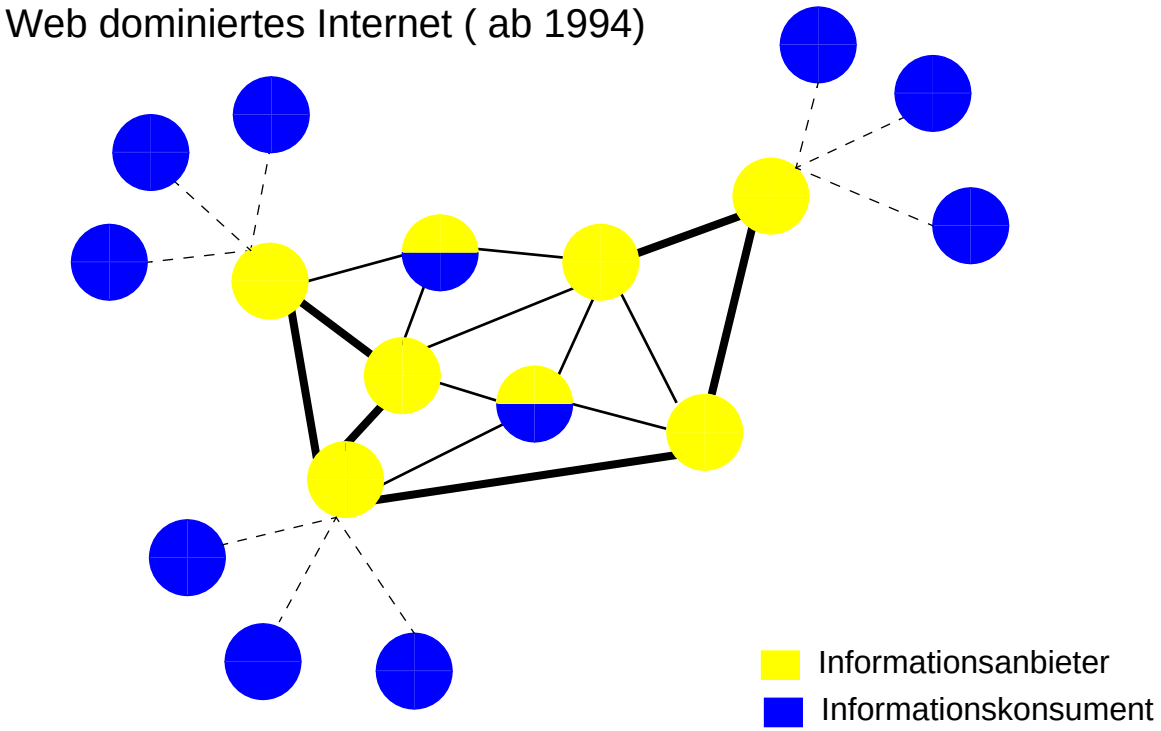
Motivation (1)

Grundform des Internet

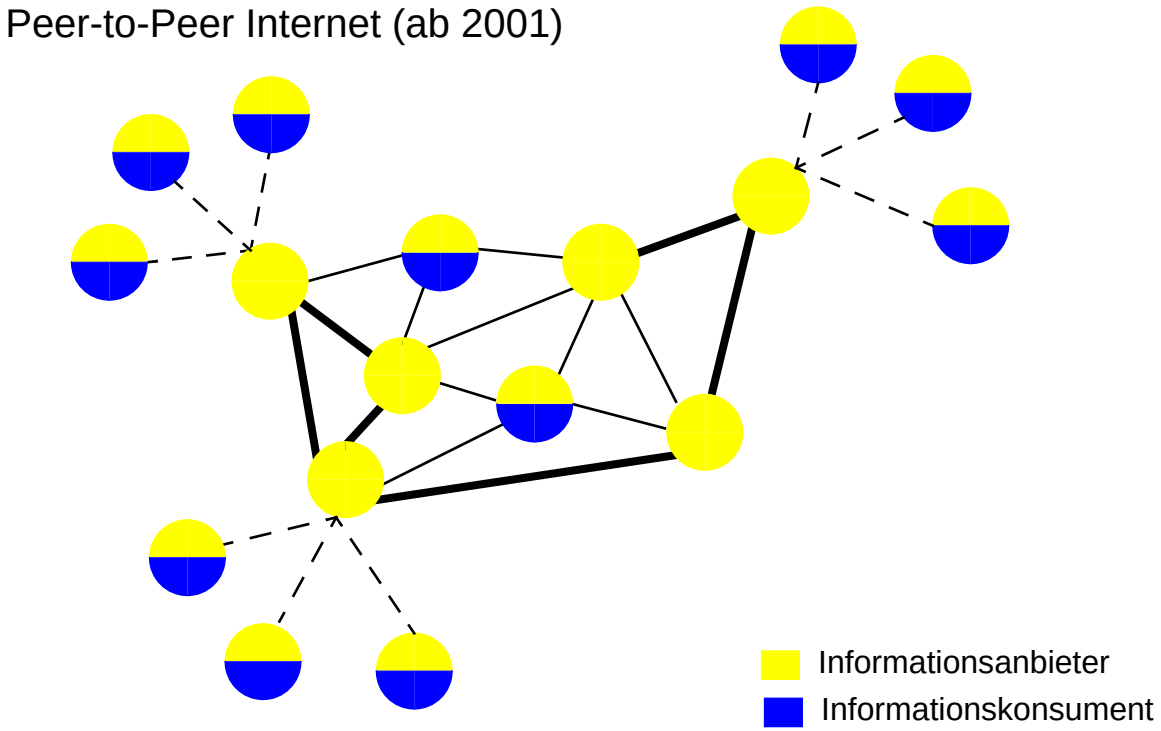


Informationsanbieter
Informationskonsument

Motivation (2)



Motivation (3)



Definition - *“hybird”* Peer-to-Peer

- “*Peer-to-Peer computing is a network-based computing model for applications where computers share resources via direct exchange between the participating computers*” (Bar01)

Definition - “*pure*” Peer-to-Peer

- Ein System kann als Peer-to-Peer bezeichnet werden wenn,
  - jedes Teilsystem sowohl Informationsanbieter als auch Informationskonsument sein kann
  - jedes Teilsystem einen gewissen Grad an Autonomie besitzt
  - die einzelnen Teilsysteme nicht permanent mit dem Gesamtsystem verbunden sein müssen
  - die einzelnen Teilsysteme keine permanente Netzwerkadresse benötigen

Übersicht: P2P-Anwendungen

- File Sharing
  - Napster, Gnutella, Morpheus Freenet, Free Haven, Publius, Mojo Nation, etc.
- Verteilte Dateisysteme
- Verteiltes Rechnen
  - SETI@home, Popular Power
- Messaging Frameworks
  - Jabber, AIMster
- Groupware

Technische Anforderungen

- Kommunikation
  - Protokolle (HTTP, SOAP)
  - NAT, Firewalls
- Lokalisierung
  - Adressierung (Nutzer, Daten)
  - Metadaten
- Stabilität
  - Fehlertoleranz
  - Verfügbarkeit von Ressourcen und Daten

Technische Anforderungen

- Sicherheit
  - Authentifikation
  - Autorisierung
- Ressourcen Verwaltung
  - Speicher
  - Bandbreite

Ziele des Seminars

- Einblick in die aktuellen Entwicklungen im Bereich Peer-to-Peer
- Vorbereitung für Studien- und Diplomarbeiten
  - wissenschaftliches Arbeiten
  - Entwicklung von Peer-to-Peer Anwendungen im Rahmen des AspectIX-Projekts