Hauptseminar: **Moderne Konzepte für weitverteilte Systeme: Peer-to-Peer-Netzwerke und fehlertolerante Algorithmen (DOOS)**

Byzantinische Fehlertoleranz durch Gruppenkommunikation am Beispiel des Rampart-Toolkit

Frank Mattauch

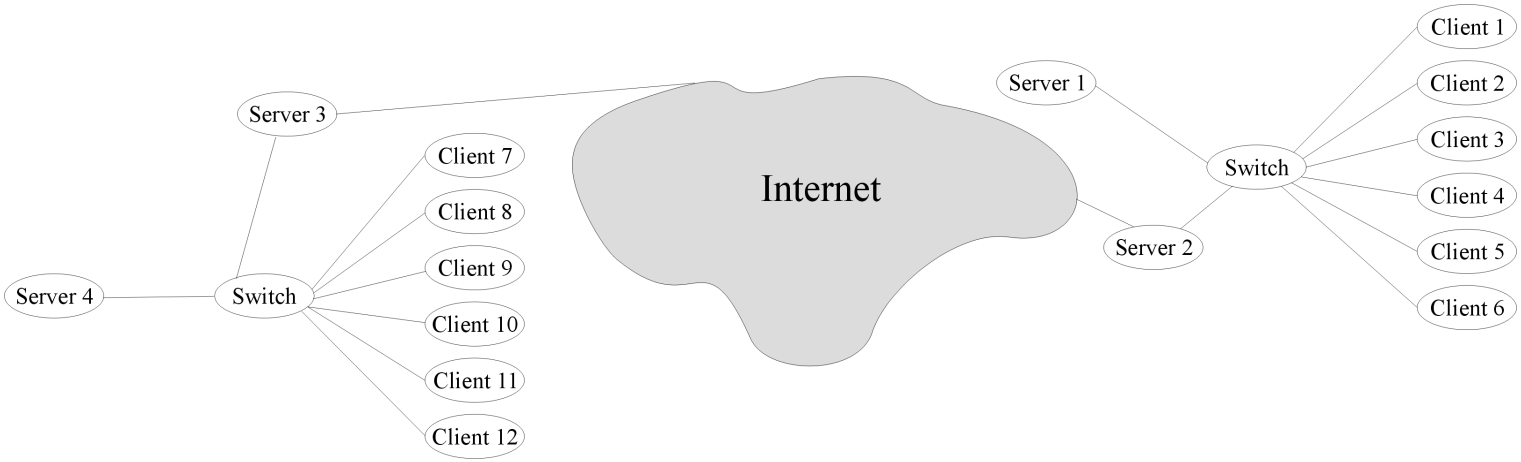
Begriffsbildung

- Rampart = Wehrgang, Wall, Schutzwall
- Toolkit = Werkzeug
- Rampart-Toolkit = “Schutschildwerkzeug” oder Werkzeug für einen Schutzschild

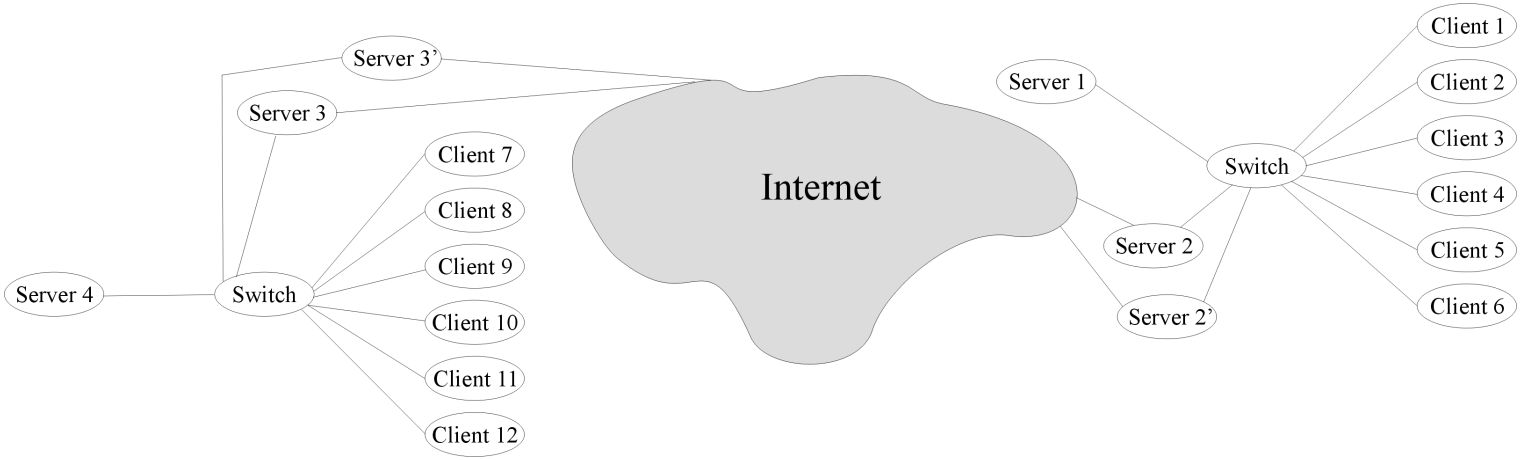
Was ist das Rampart-Toolkit?

- Kern besteht aus Protokollen
- Byzantinische Fehlertoleranz bei
  - zuverlässigem und
  - atomaren Gruppenmulticast
- Erstes System, das dies demonstrierte

Verteiltes System ohne Redundanz



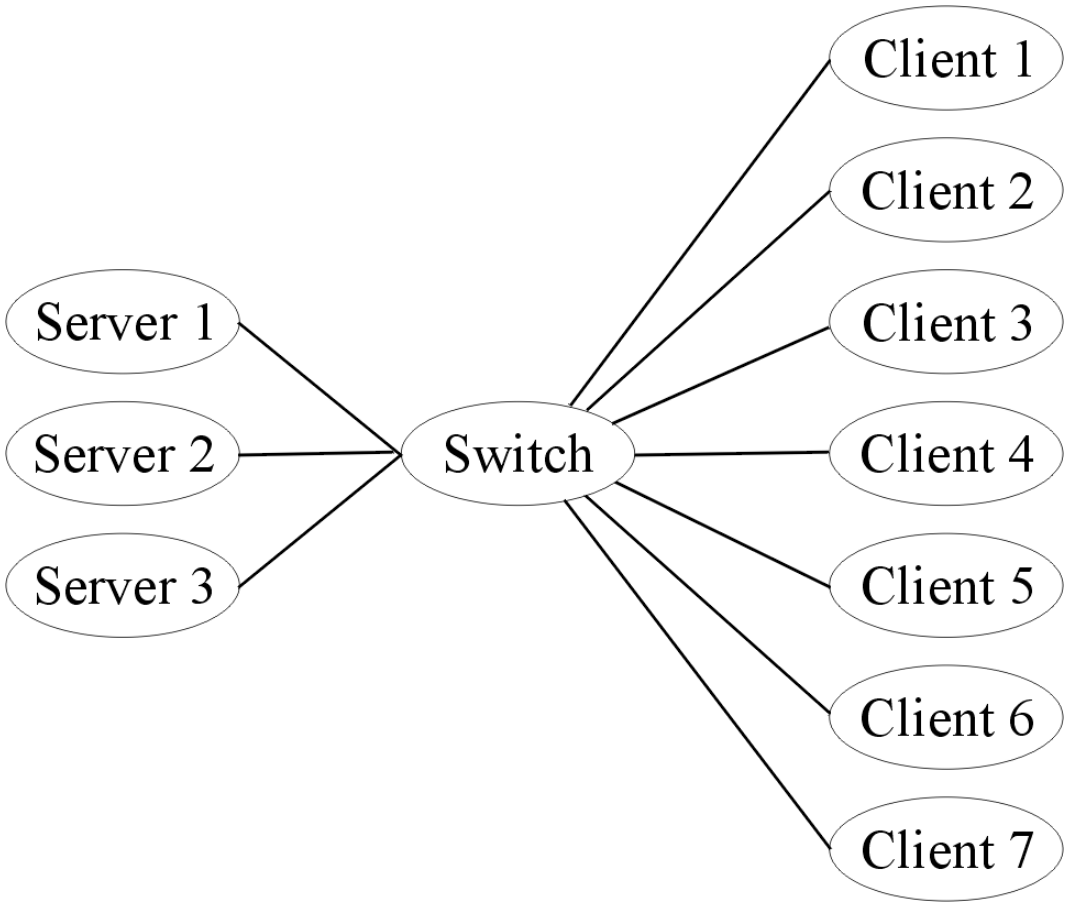
Verteiltes System mit Redundanz



State Machine Replication - Eigenschaften

- Dienst implementiert durch mehrere Server.
- Eigenschaften der Server:
  - Identisch
  - Deterministisch
  - Haben selben Anfangszustand

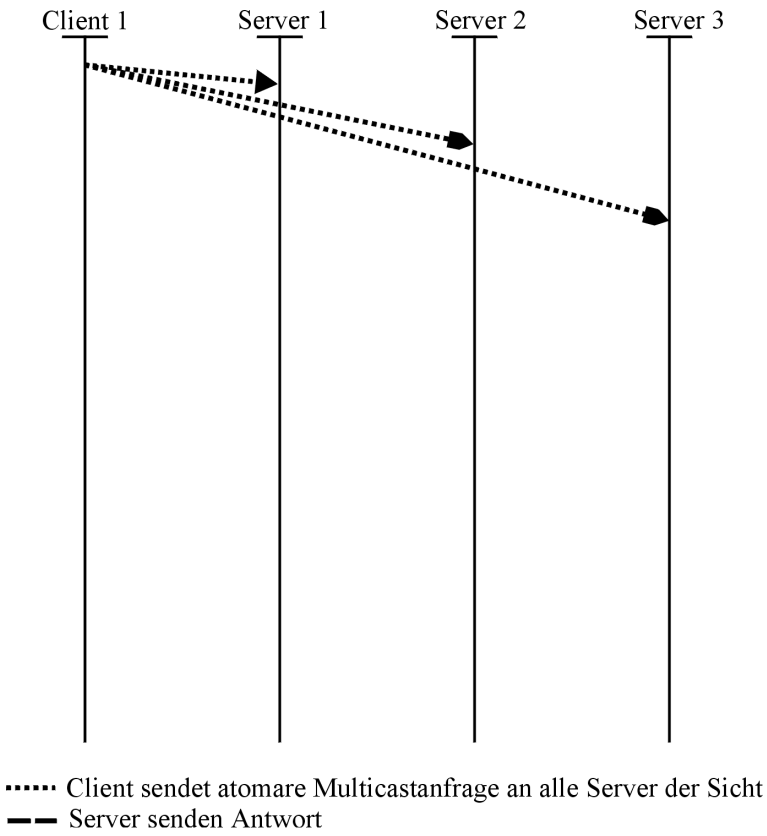
State Machine Replication - Szenario



State Machine Replication - Ablauf

1. Anfragen von Clients mittels atomarem Multicast-protokoll

State Machine Replication - Ablauf

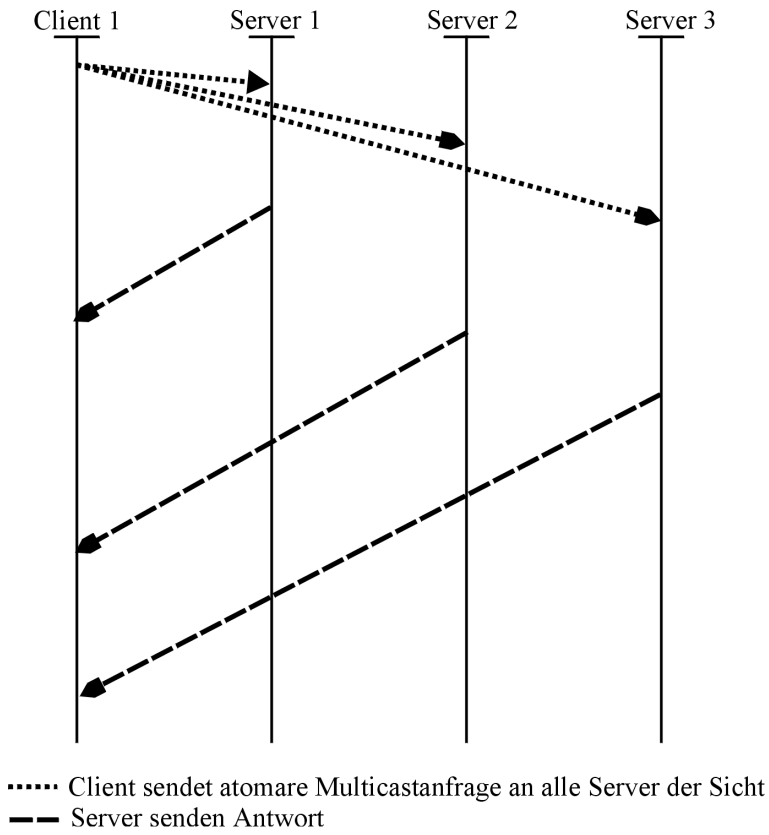


State Machine Replication - Ablauf

1. Anfragen von Clients mittels atomarem Multicast-protokoll

2. Reihenfolge erhaltende Bearbeitung bei den Servern

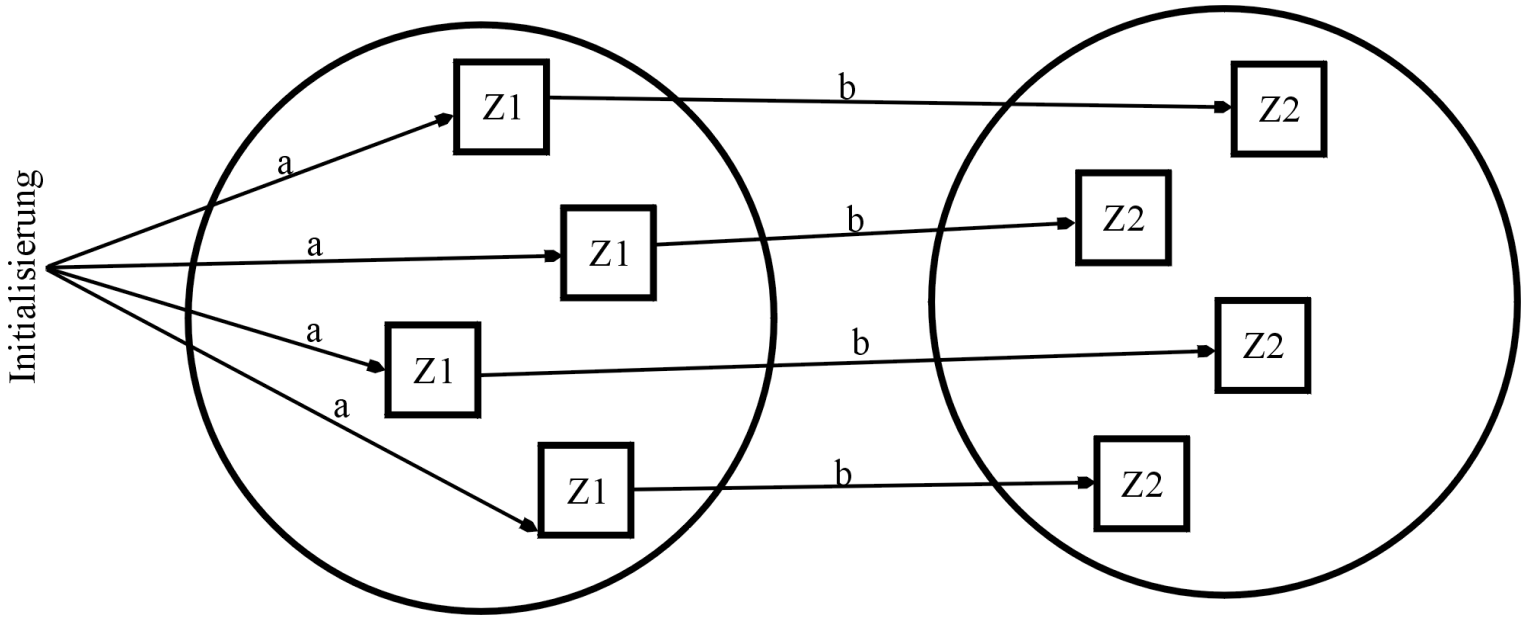
State Machine Replication - Ablauf



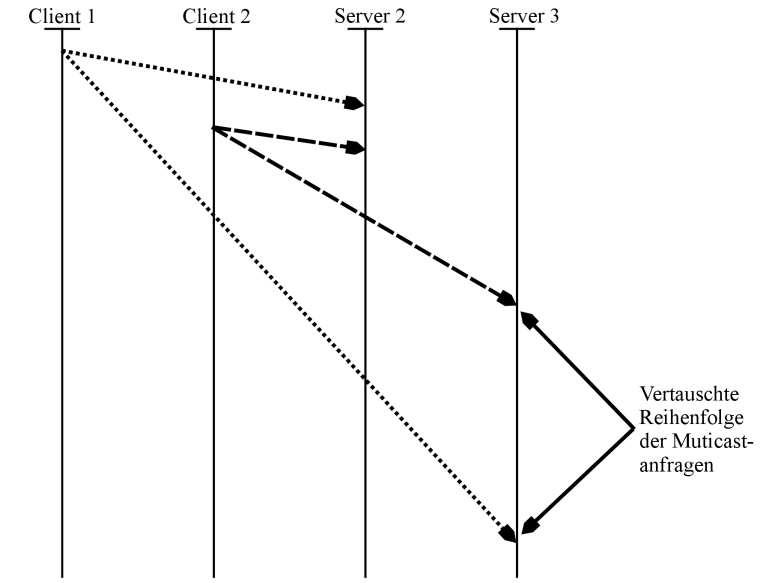
State Machine Replication - Ablauf

1. Anfragen von Clients mittels atomarem Multicast-protokoll
2. Reihenfolge erhaltende Bearbeitung bei den Servern
3. Output-Voting beim Client

State Machine Replication - Ablauf



State Machine Replication - Reihenfolgeerhaltung



Client 1 sendet atomare Multicastanfrage an alle Server der Sicht
Client 2 sendet eine weitere atomare Multicastanfrage
In diesem Bild sind die Antworten der Server zwecks Übersichtlichkeit nicht abgebildet.

Annahmen zur Umgebung der Protokolle von Rampart

- Verwendung digitaler Signaturen
- Private Schlüssel bei Severprozessen, öffentliche bei allen anderen
- Erste Schlüsselverteilung kann manuell erfolgen
- Öffentliche Schlüssel sind nach einem Ausfall wieder herstellbar

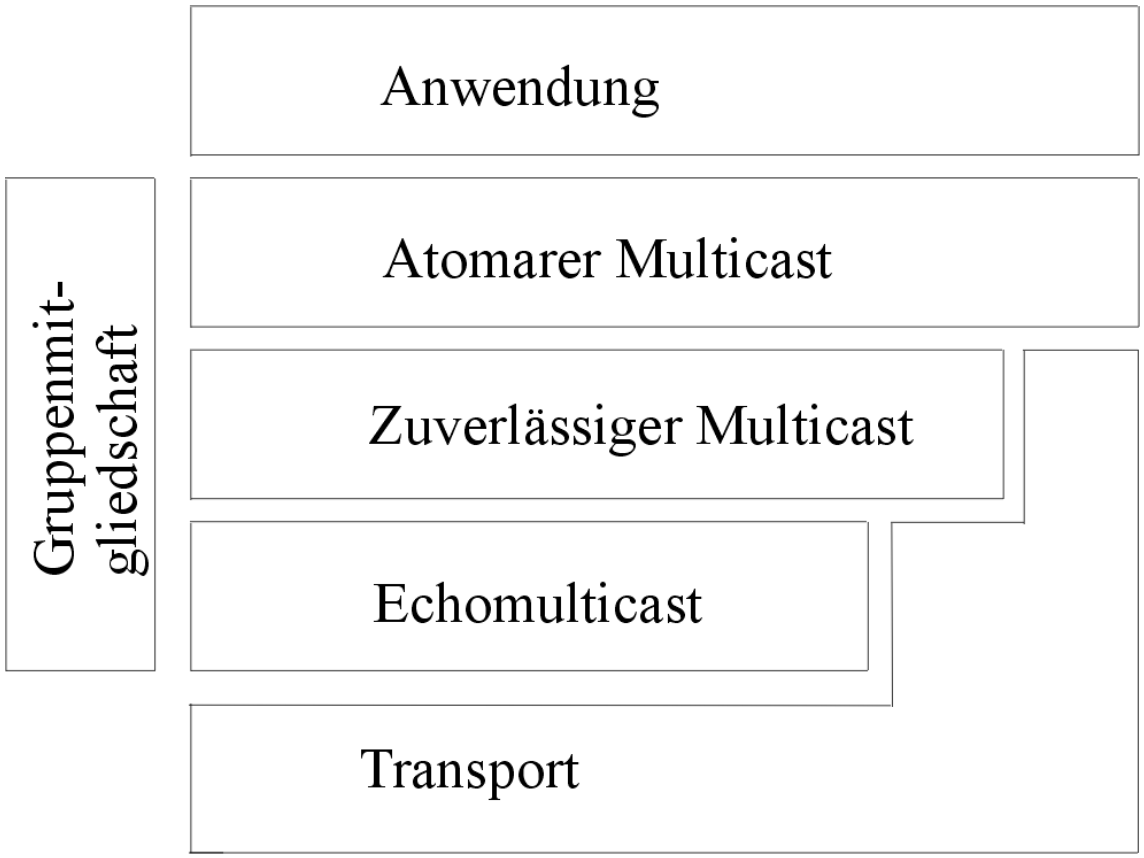
Annahmen zur Umgebung der Protokolle von Rampart

- Schlüssel eines neuen Servers wird verteilt, bevor dieser in Aktion tritt
- Existenz eines Kommunikationskanals
  - Zuverlässig
  - Authentifizierbar
  - Punkt-zu-Punktverbindung

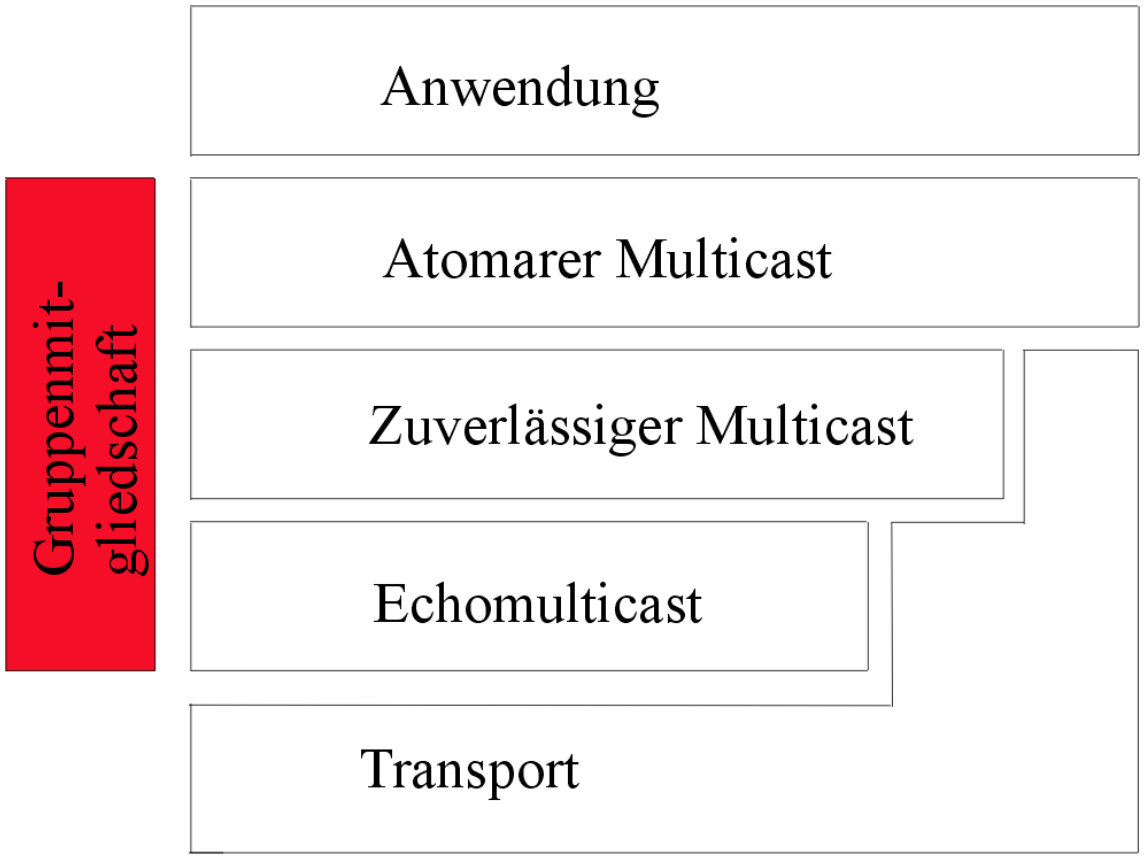
Annahmen zur Umgebung der Protokolle von Rampart

- Keine Annahmen zur Art der Fehler, da byzantinische Fehler
- Weniger als ein Drittel der Gruppenmitglieder fehlerhaft
- Gesamte System ist asynchron

Protokollschichten des Rampart-Toolkit



Gruppenmitgliedschaftsprotokoll

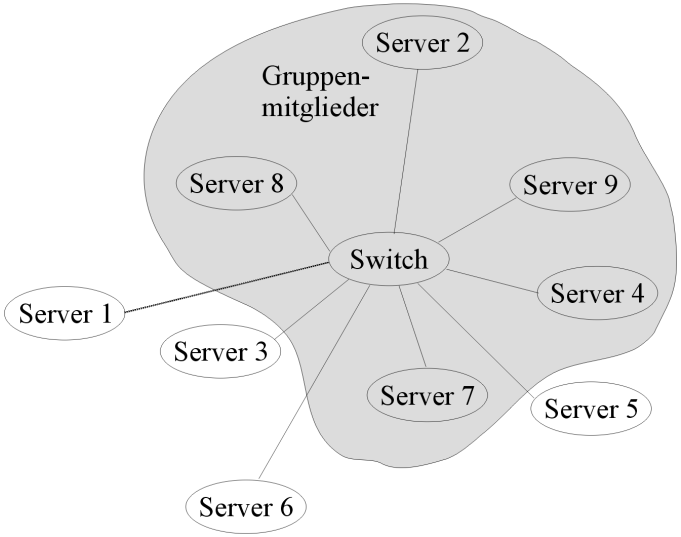


Gruppenmitgliedschaftsprotokoll - Aufgaben

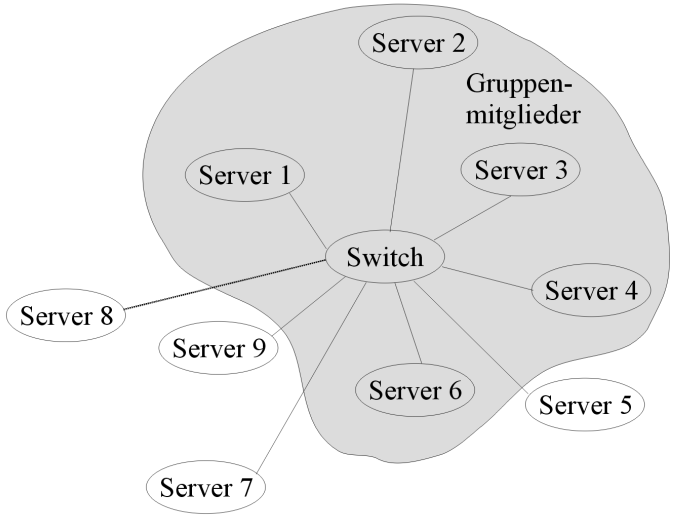
- Erstellen von Gruppensichten
- Ausliefern neuer Gruppensichten
- Hinzufügen und Entfernen von Gruppenmitgliedern
- Für Änderungen ausreichende Mehrheit notwendig

Gruppenmitgliedschaftsprotokoll - Gruppensichten

Unterschiedliche Gruppensichten



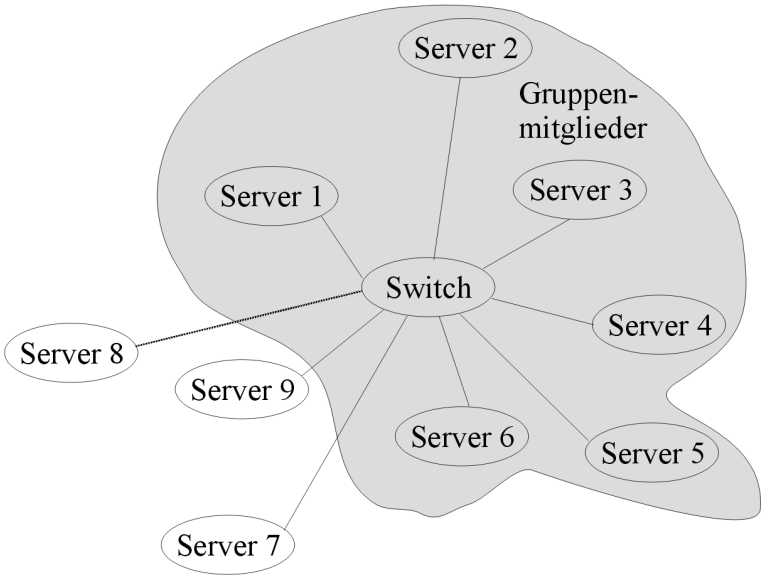
Gruppensicht 1



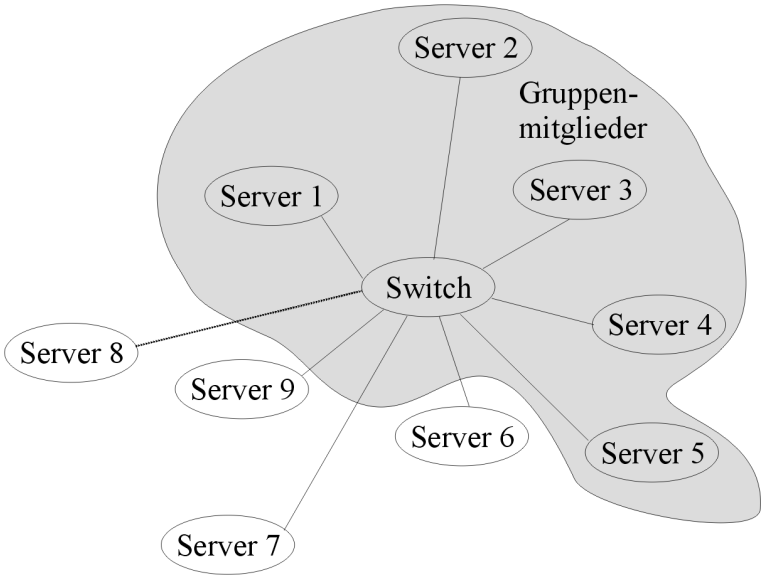
Gruppensicht 2

Gruppenmitgliedschaftsprotokoll - Gruppensichten

Hinzufügen eines Elements zur Gruppensicht 2



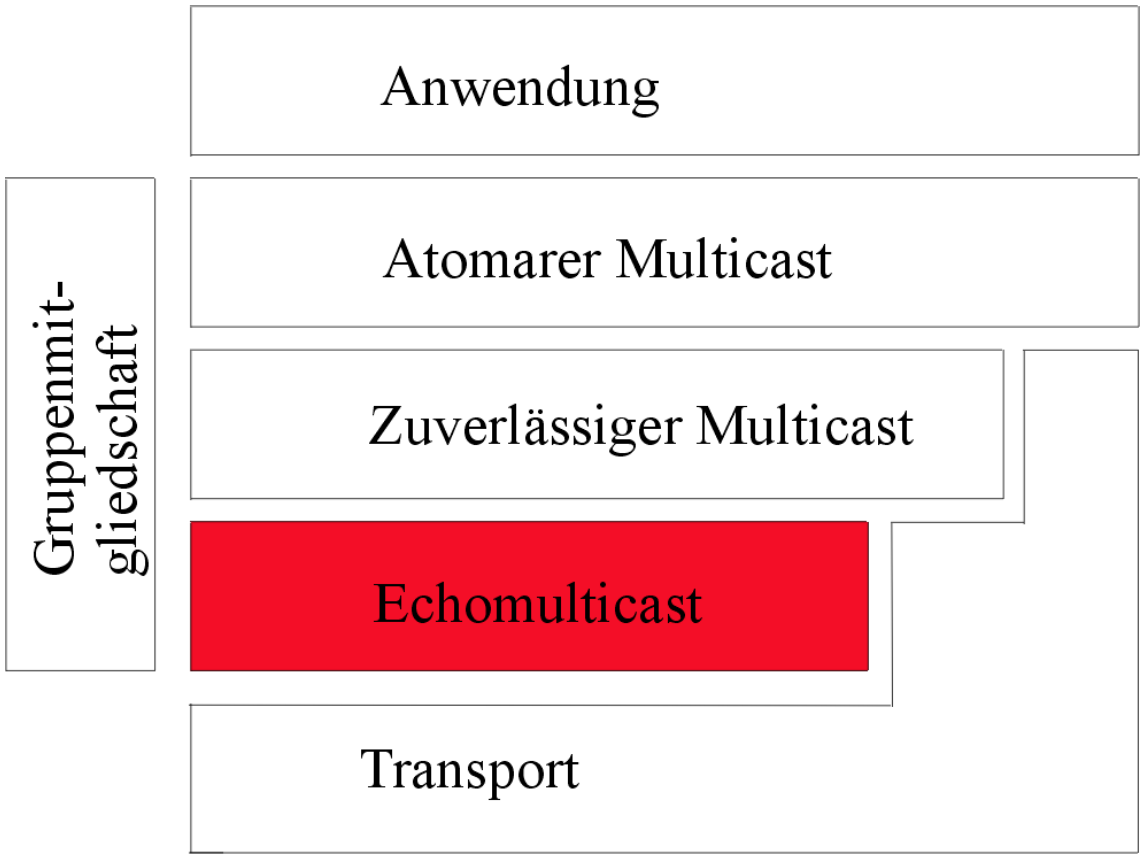
Löschen eines Elements aus der Gruppensicht 2



Gruppenmitgliedschaftsprotokoll - Gruppensichten

- Böswillige Änderungen einer Gruppensicht nur durch Denial-of-Service Angriffe
- Evtl. keine Einigung auf eine Gruppensicht möglich, wenn alle Mitglieder zu schnell wechseln
- Kostenintensiv durch RSA

Echomulticast



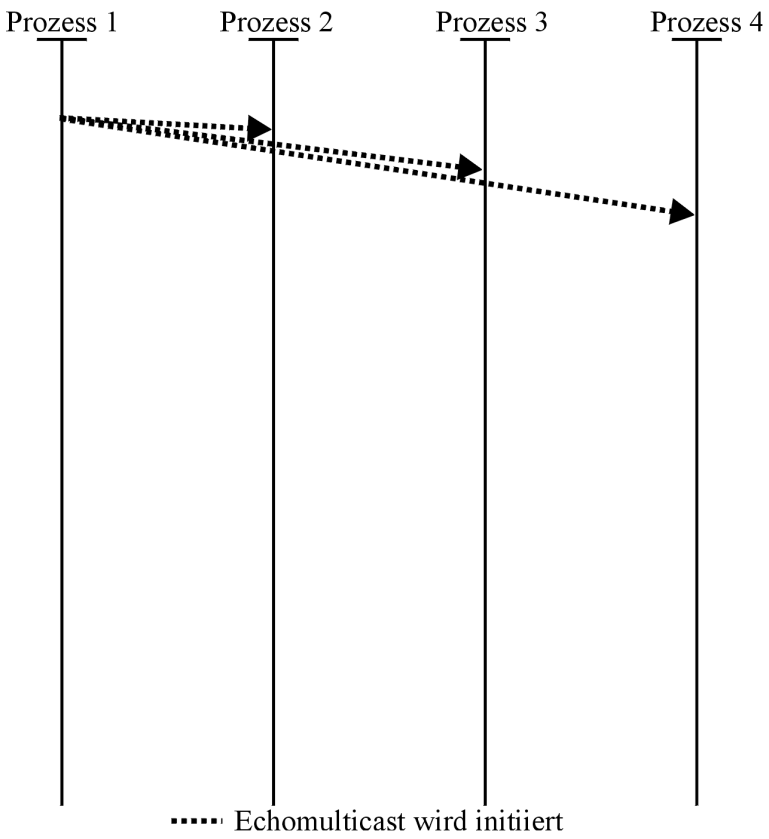
Echomulticast - Aufgaben

- Sicherstellen, dass jeder korrekte Prozess einer Gruppensicht dieselbe Nachricht an die nächsthöhere Schicht, den zuverlässigen Multicast, liefert

Echomulticast - Ablauf

1. Multicasten einer Nachricht

Echomulticast - Ablauf

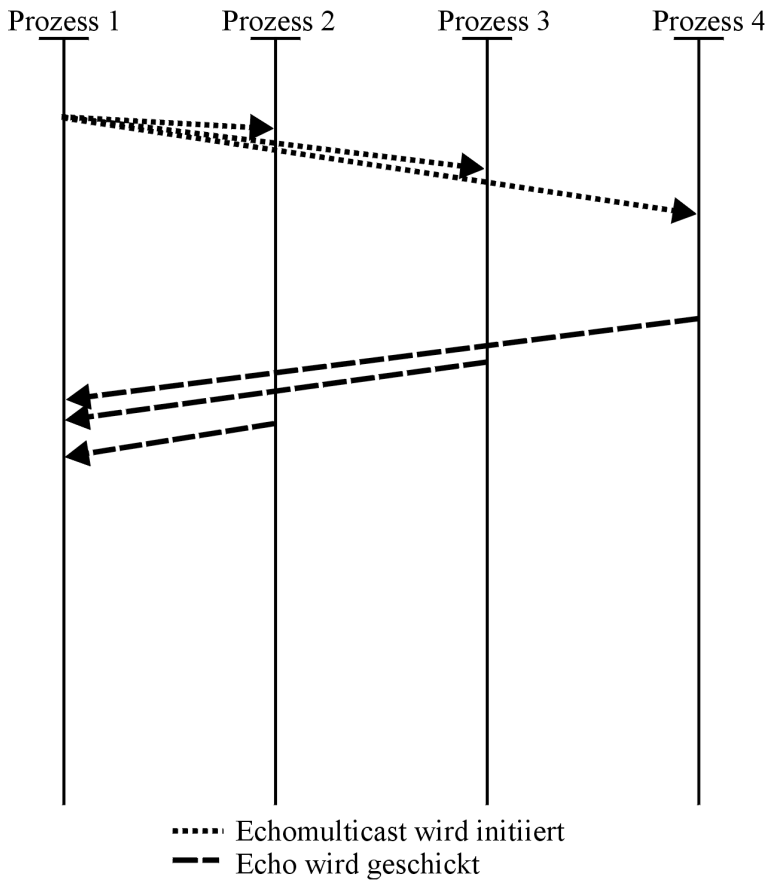


Echomulticast - Ablauf

1. Multicasten einer Nachricht

2. Signieren der Nachricht und Antwort als Echo senden

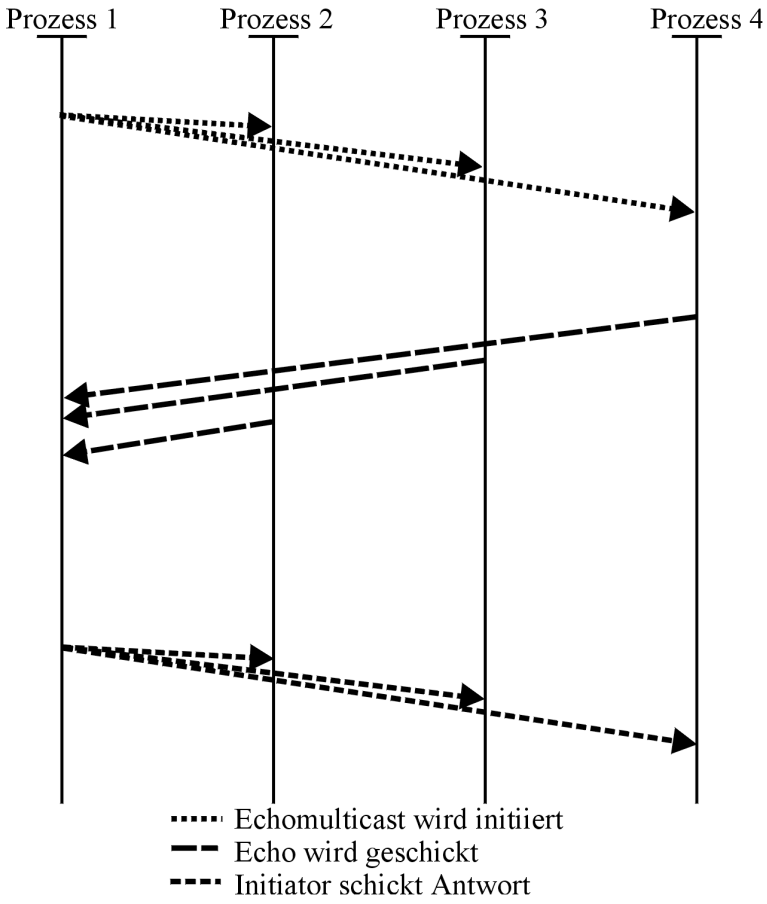
Echomulticast - Ablauf



Echomulticast - Ablauf

1. Multicasten einer Nachricht
2. Signieren der Nachricht und Antwort als Echo senden
3. Warten auf Echos und senden der Echos als Paket an die Gruppe

Echomulticast -Ablauf



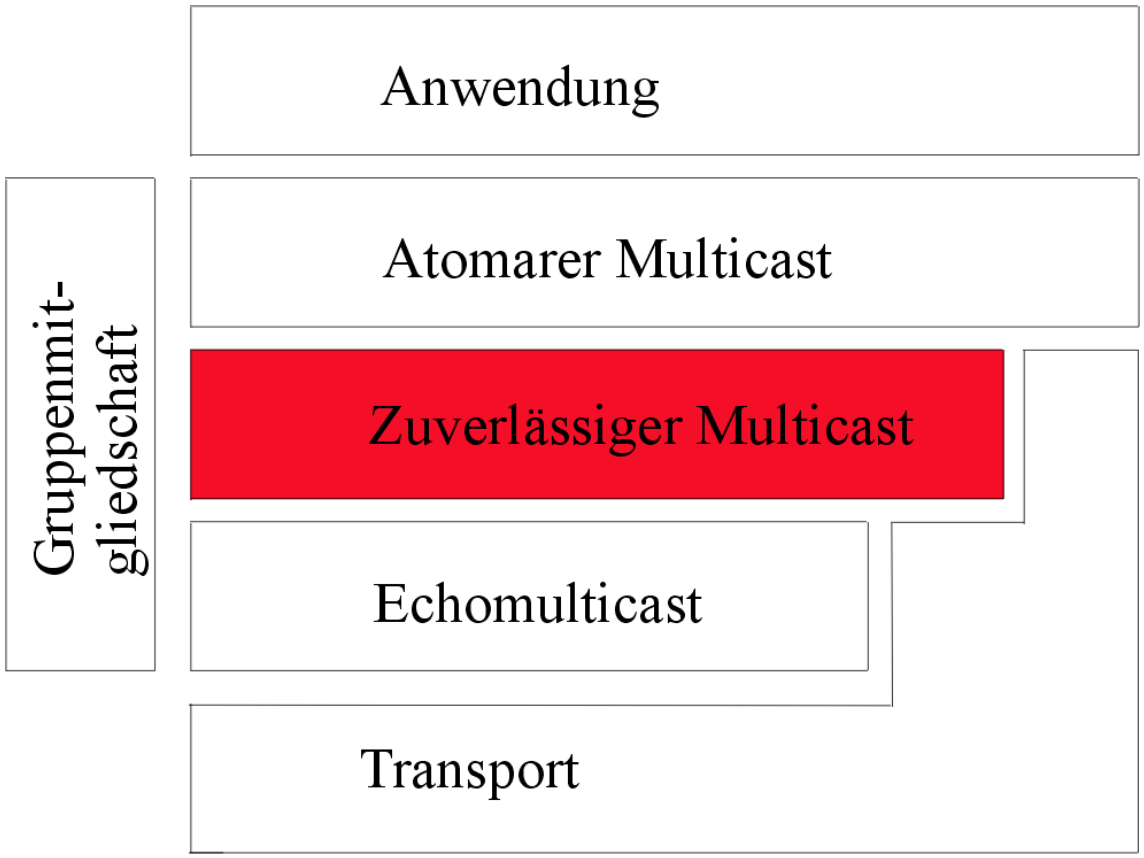
Echomulticast - Ablauf

1. Multicasten einer Nachricht
2. Signieren der Nachricht und Antwort als Echo senden
3. Warten auf Echos und senden der Echos als Paket an die Gruppe
4. Überprüfen auf
   - ❑ korrekte Signaturen
   - ❑ selbe Sichtnummer

Echomulticast - Ablauf

5. Falls Überprüfung erfüllt: Weitergabe der Nachricht

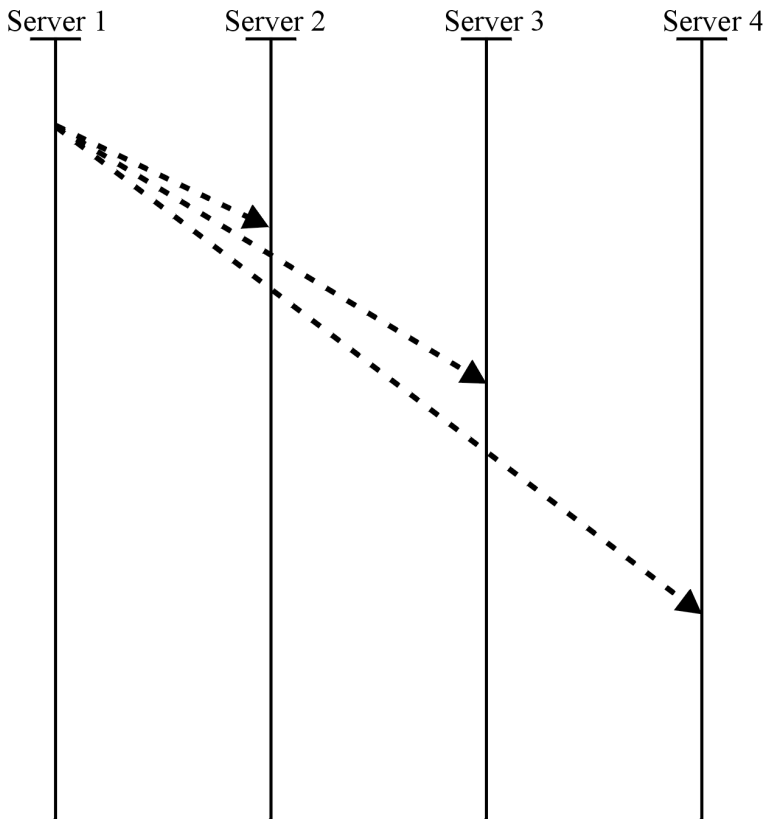
Zuverlässiger Multicast



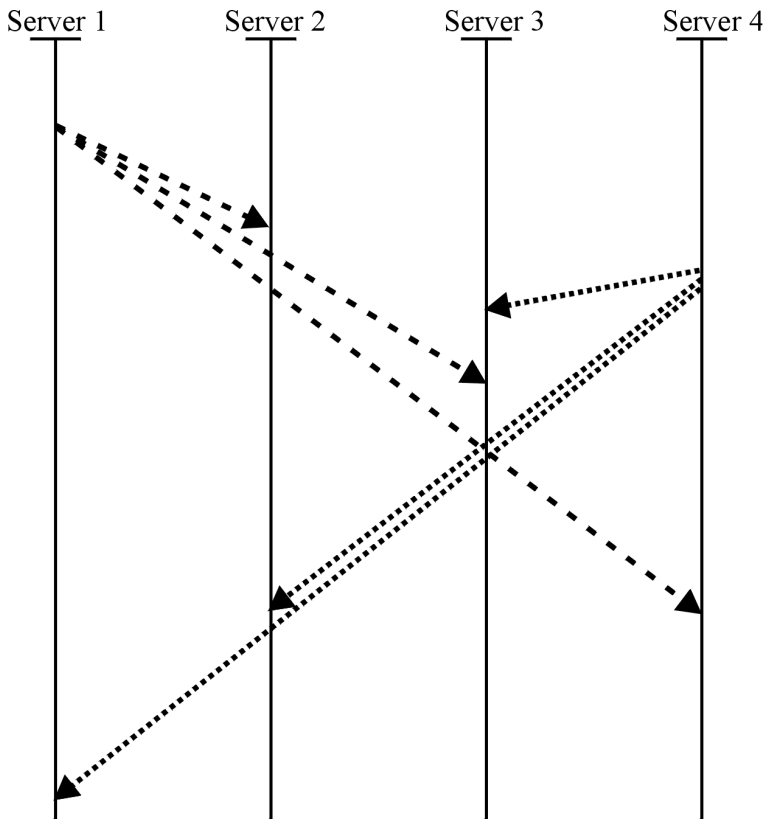
Zuverlässiger Multicast - Aufgaben

- Alle korrekten Gruppenmitglieder bekommen dieselben Nachrichten trotz böswilliger Multicasts manipulierter Mitlieder (ohne Reihenfolge!)
- Liefern von Gruppensichten

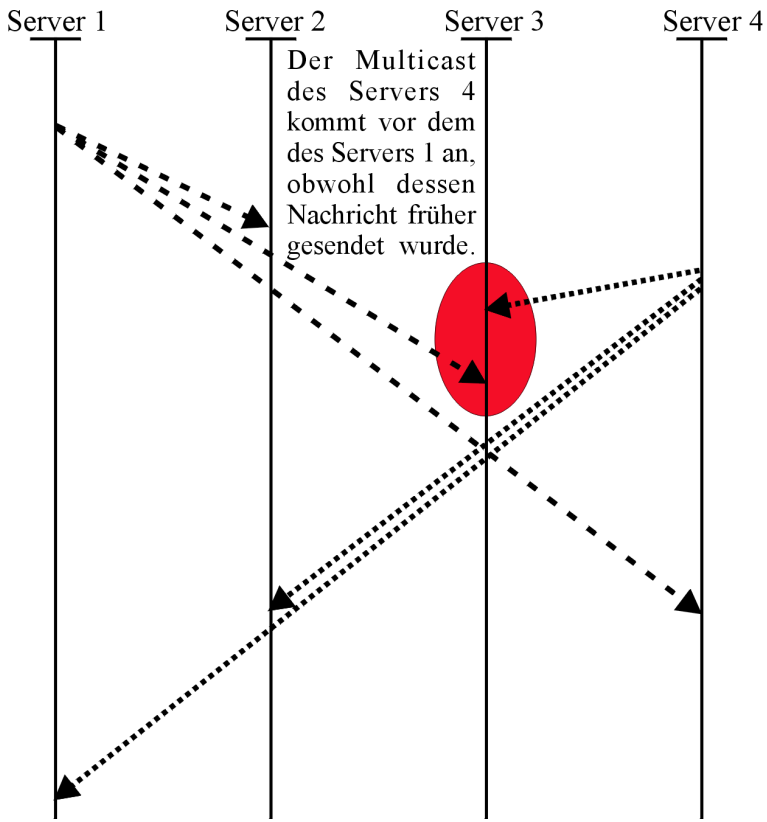
Zuverlässiger Multicast - Eigenschaften



Zuverlässiger Multicast - Eigenschaften



Zuverlässiger Multicast - Eigenschaften



Zuverlässiger Multicast - Eigenschaften

- Nachricht ist
  - Gruppensicht
  - Multicastnachricht eines Gruppenmitglieds
- Fortschritt des Protokolls kann von der Entfernung eines Guppenmitglieds abhängen

Zuverlässiger Multicast - Ablauf

- Zwei Fälle:

  1. Keine Änderung der Gruppenzugehörigkeit:
     - Zuverlässiger Multicast führt Echomulticast aus
     - Reicht Nachrichten von Echomulticastschicht einfach weiter
  2. Änderung der Gruppenzugehörigkeit

Zuverlässiger Multicast - Ablauf

- Änderung der Gruppenzugehörigkeit:
  Zwei Fälle:

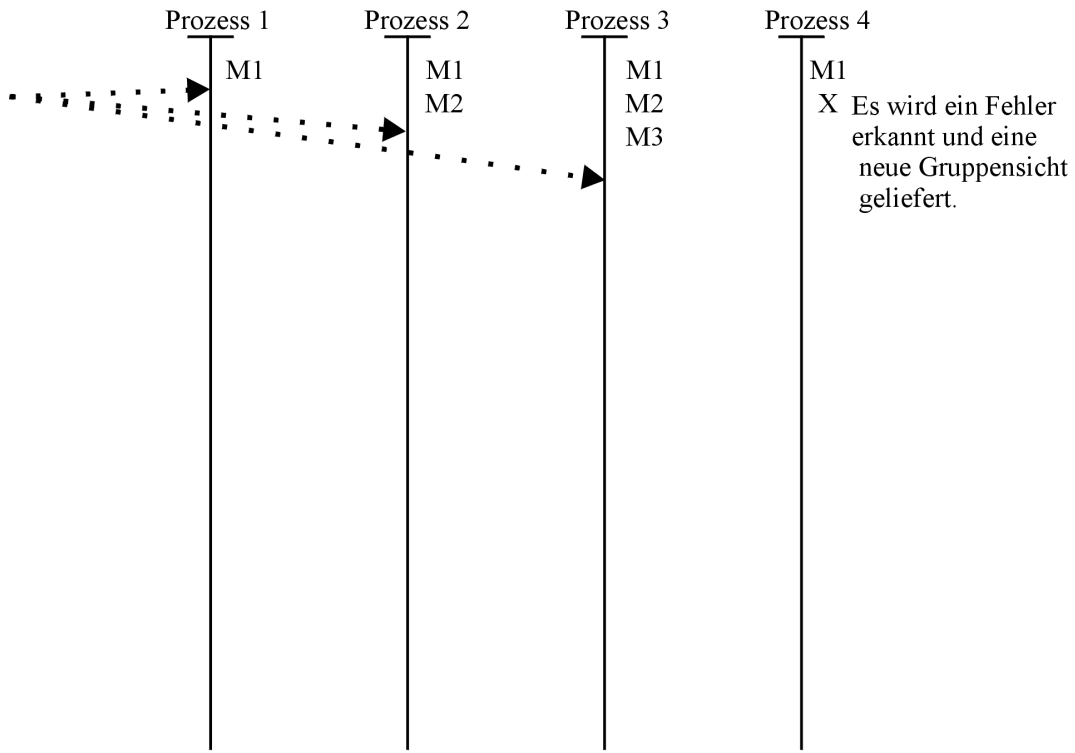
  1. Keine weiter Änderung der Gruppenzugehörigkeit während eines Protokolldurchlaufs
  2. Änderungen sind notwendig (z.B. “Flush” oder “Ende”-Nachricht wird vom Prozess nie empfangen)

Zuverlässiger Multicast - Ablauf

- Änderung der Gruppenzugehörigkeit und keine weitere Änderung während eines Protokolldurchlaufs:

  1. Multicast einer neuen Gruppensicht

Zuverlässiger Multicast - Ablauf



Mx Prozess hat die Nachricht x bekommen
X Prozess verhält sich nicht mehr regelkonform (d.h. er bleibt hängen, wird manipuliert, usw ...)

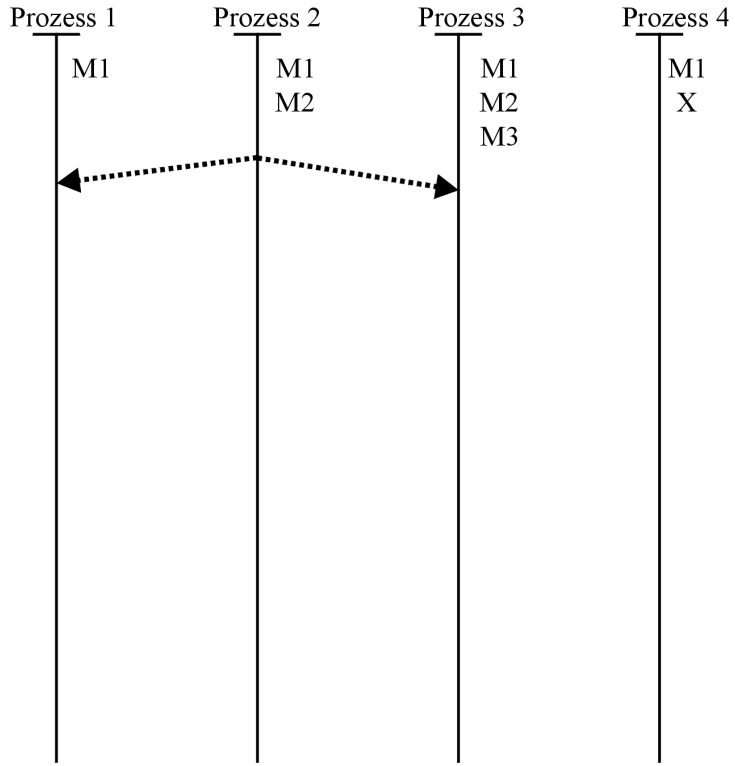
Zuverlässiger Multicast - Ablauf

- Änderung der Gruppenzugehörigkeit und keine weitere während eines Protokolldurchlaufs:

  1. Multicast einer neuen Gruppensicht

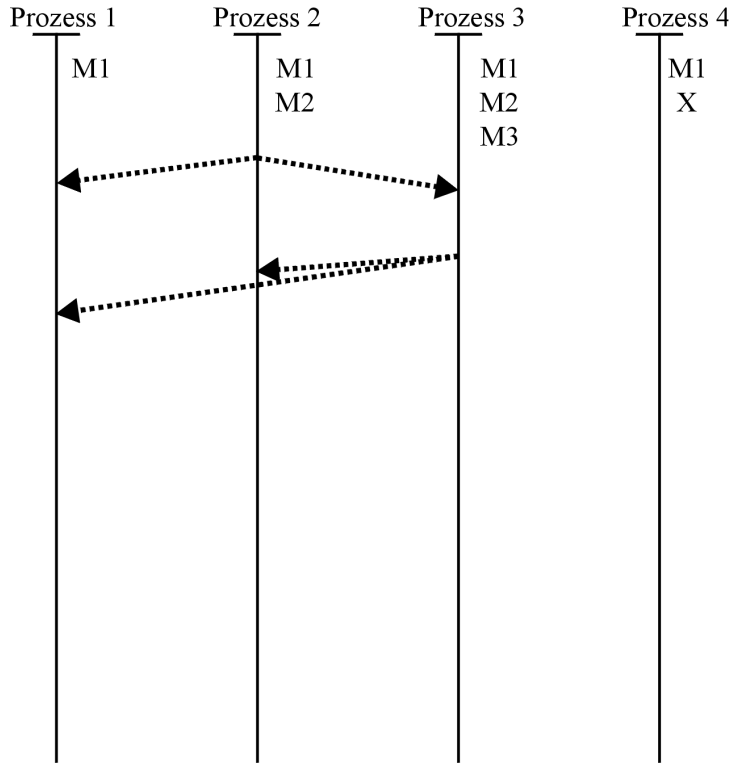
  2. Bei Empfang:
     - Prozess unterlässt zuverlässige Multicasts
     - "Ende"-Nachricht senden
     - Auf "Ende-Nachrichten der anderen warten

Zuverlässiger Multicast - Ablauf



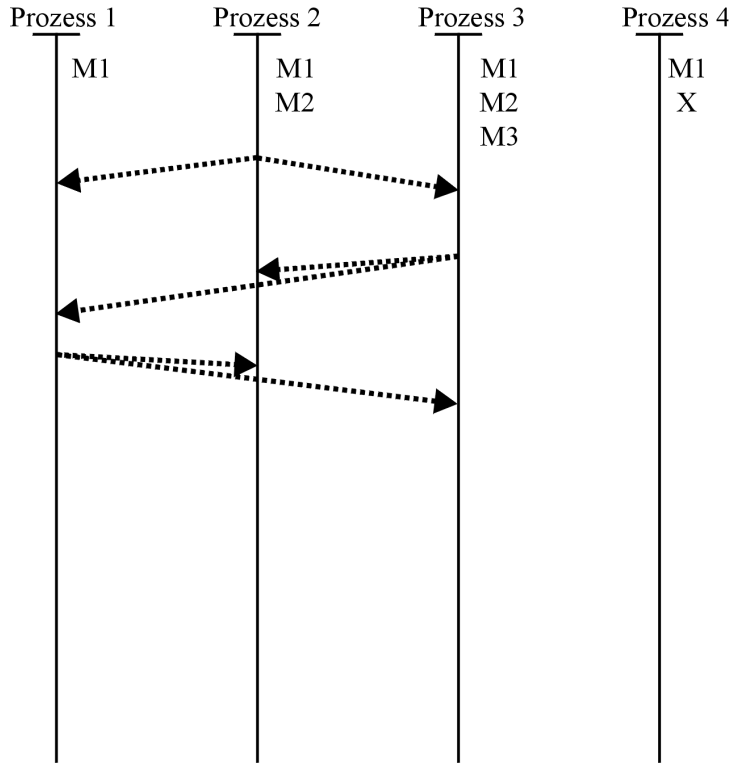
Mx Prozess hat die Nachricht x bekommen
Prozess sendet die Nachricht “Ende”
X Prozess verhält sich nicht mehr regelkonform
(d.h. er bleibt hängen, wird manipuliert, usw ...)

Zuverlässiger Multicast - Ablauf



Mx Prozess hat die Nachricht x bekommen
Prozess sendet die Nachricht "Ende"
X Prozess verhält sich nicht mehr regelkonform
(d.h. er bleibt hängen, wird manipuliert, usw ...)

Zuverlässiger Multicast - Ablauf



Mx Prozess hat die Nachricht x bekommen
Prozess sendet die Nachricht "Ende"
X Prozess verhält sich nicht mehr regelkonform
(d.h. er bleibt hängen, wird manipuliert, usw ...)

Zuverlässiger Multicast - Ablauf

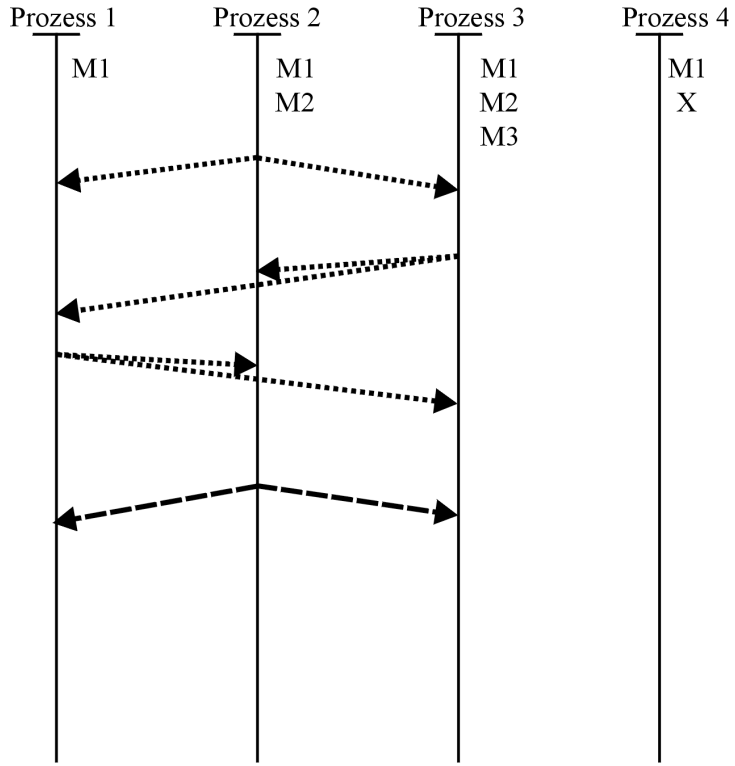
- Änderung der Gruppenzugehörigkeit und keine Änderung während eines Protokolldurchlaufs

  1. Multicast einer neuen Gruppensicht
  2. Bei Empfang:
     - Prozess unterlässt zuverlässige Multicasts
     - "Ende"-Nachricht senden
     - Auf "Ende-Nachrichten der anderen warten

Zuverlässiger Multicast - Ablauf

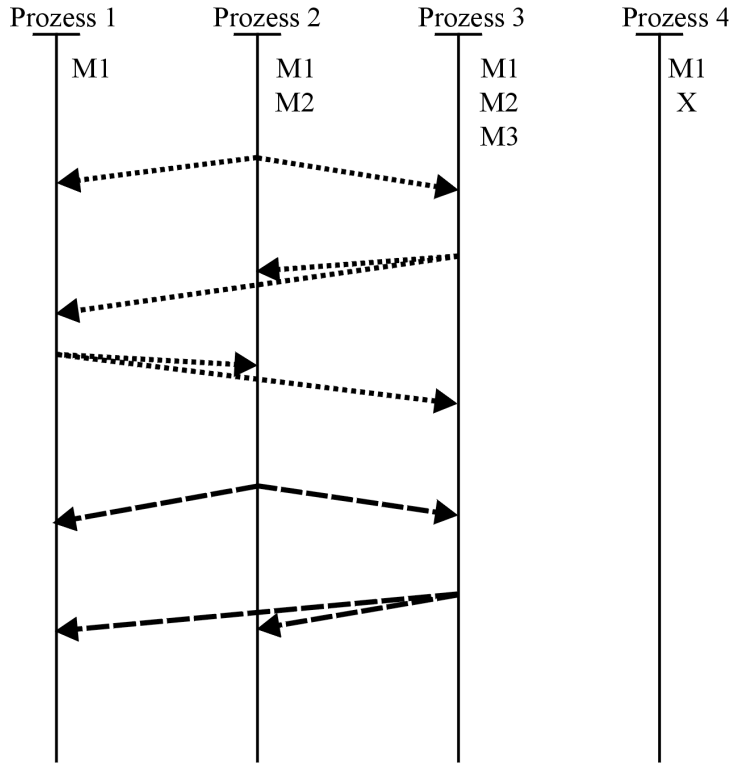
3. Nach Empfang aller "Ende"-Nachrichten:
   - "Flush"-Nachricht senden
   - Auf "Flush"-Nachrichten der anderen warten
4. Weiterreichen der Sicht an die atomare Multicast-schicht
5. Wiederaufnahme des zuverlässigen Multicasts

Zuverlässiger Multicast - Ablauf



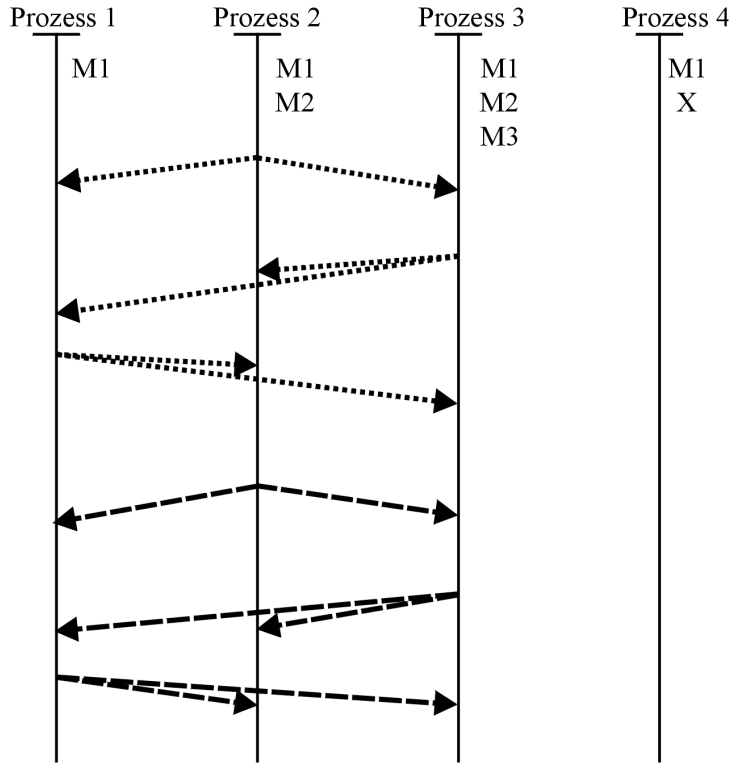
Mx Prozess hat die Nachricht x bekommen
Prozess sendet die Nachricht "Ende"
Prozess sendet Flushnachricht
X Prozess verhält sich nicht mehr regelkonform
(d.h. er bleibt hängen, wird manipuliert, usw ...)

Zuverlässiger Multicast - Ablauf



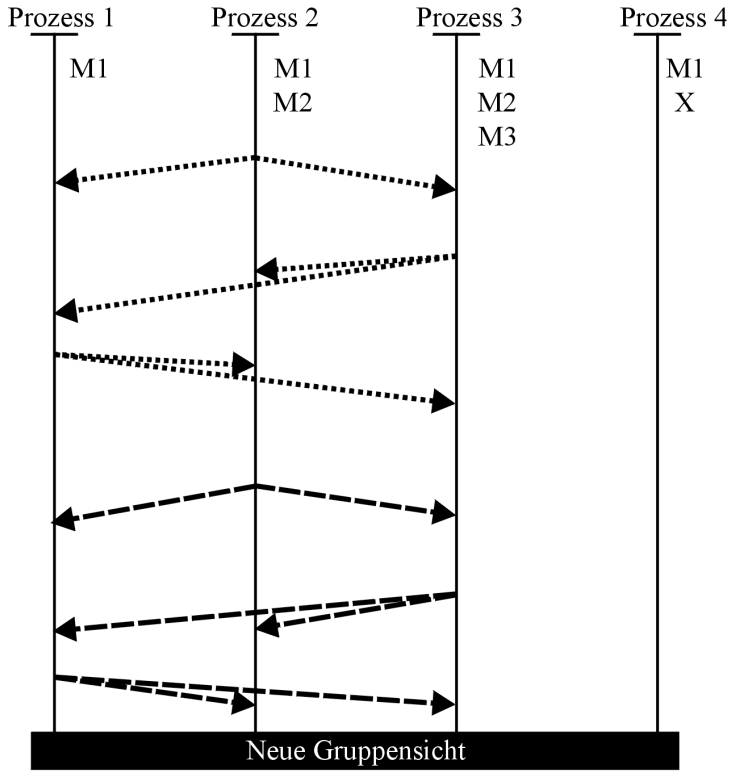
Mx Prozess hat die Nachricht x bekommen
Prozess sendet die Nachricht "Ende"
Prozess sendet Flushnachricht
X Prozess verhält sich nicht mehr regelkonform
(d.h. er bleibt hängen, wird manipuliert, usw ...)

Zuverlässiger Multicast - Ablauf



Mx Prozess hat die Nachricht x bekommen
Prozess sendet die Nachricht "Ende"
Prozess sendet Flushnachricht
X Prozess verhält sich nicht mehr regelkonform
(d.h. er bleibt hängen, wird manipuliert, usw ...)

Zuverlässiger Multicast - Ablauf



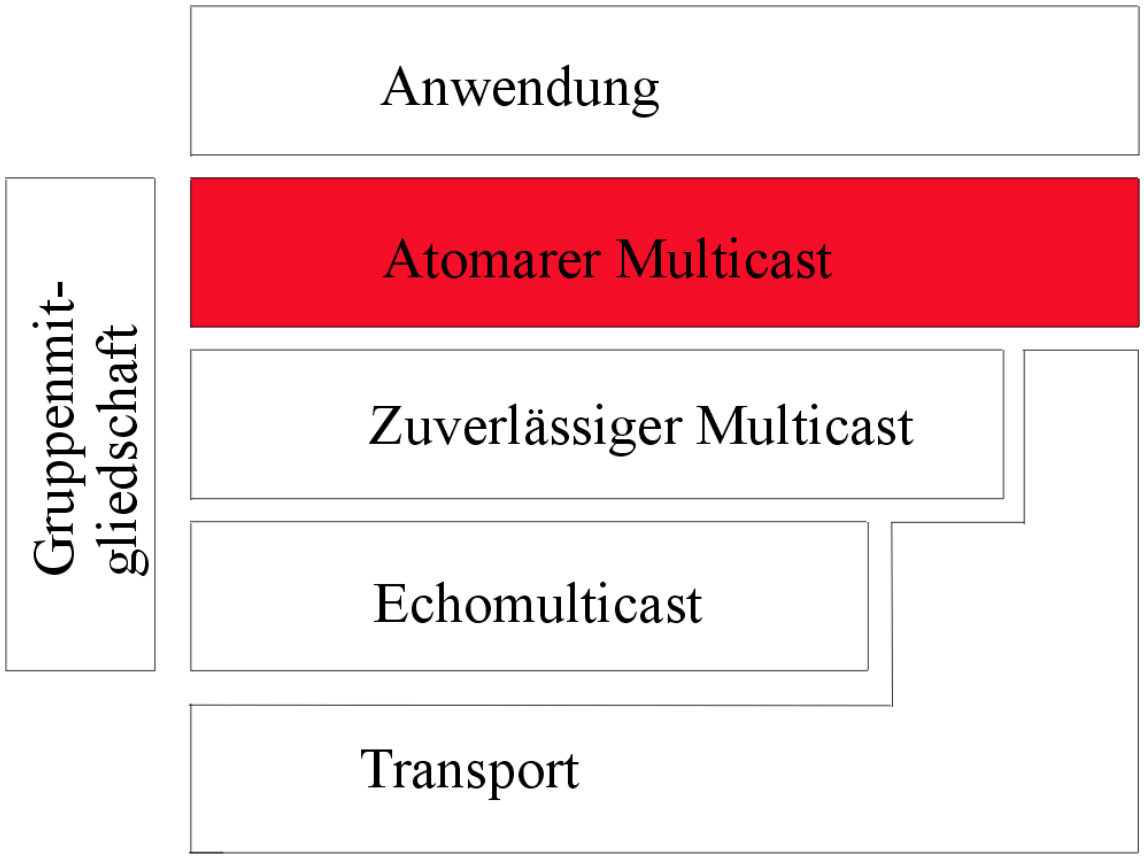
Zuverlässiger Multicast - Ablauf

- Änderung der Gruppenmitgliedschaft und Änderung während eines Protokolldurchlaufs:
  1. Empfang einer neuen Sicht
  2. Keine neuen Prozesse mehr in Gruppensicht aufnehmen
  3. Nicht reagierende Mitglieder entfernen und neue Sicht erstellen

Zuverlässiger Multicast - Ablauf

4. Wiederhole bis Weitergabe von Verwaltungsinformation und Gruppensicht an atomare Multicastschicht möglich

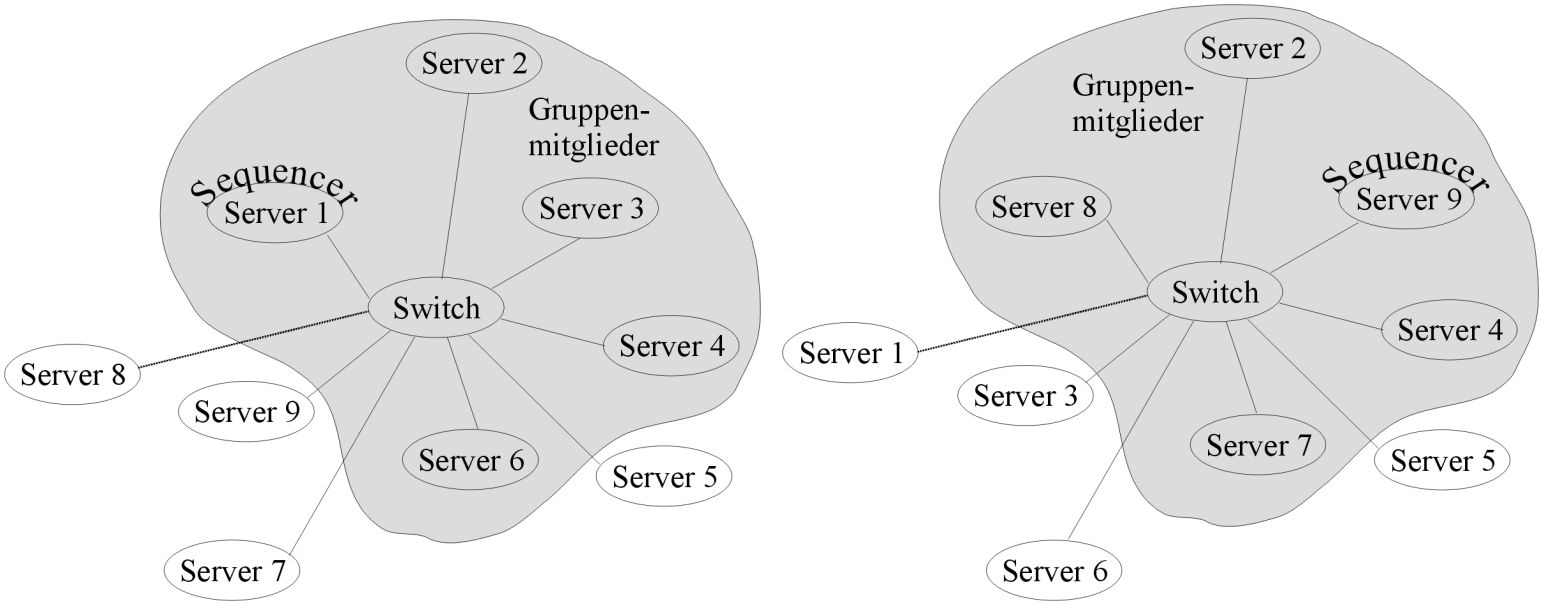
Atomarer Multicast



Atomarer Multicast - Aufgaben

- Alle korrekten Gruppenmitglieder liefern dieselbe Nachricht in derselben Reihenfolge (Unterschied zuverlässiger Multicast!)

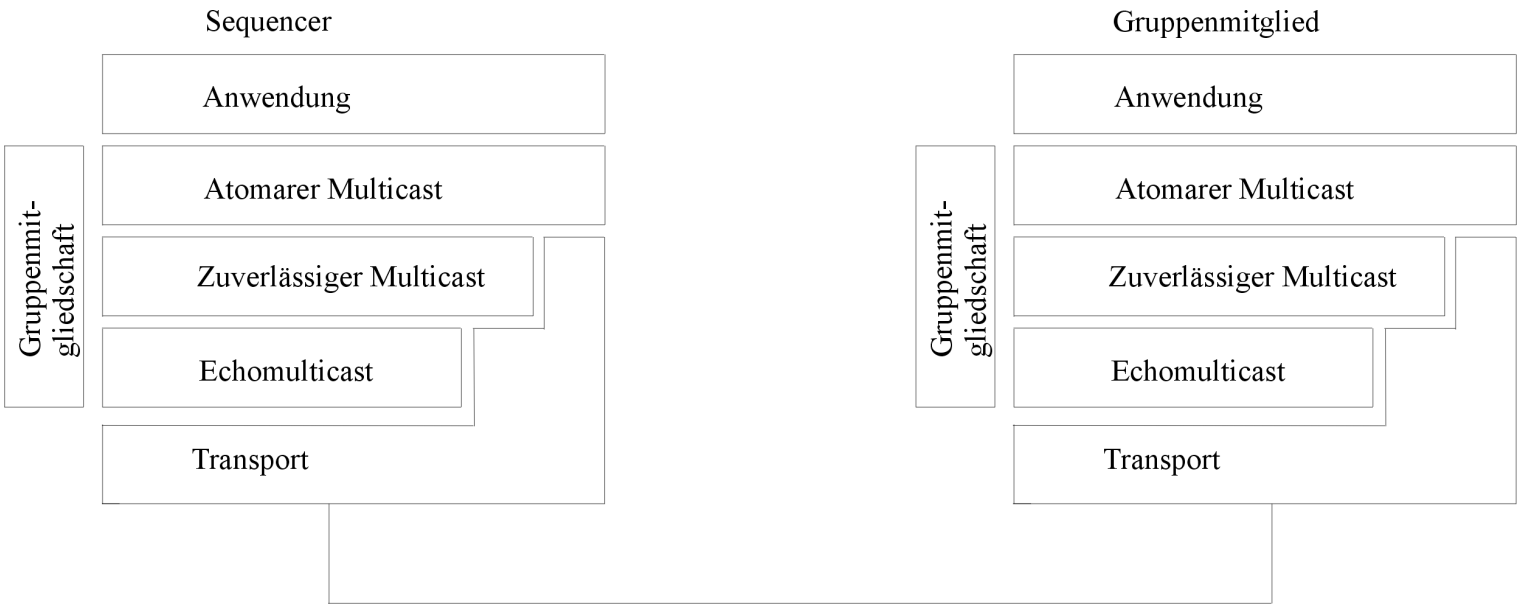
Atomarer Multicast - Sequencer



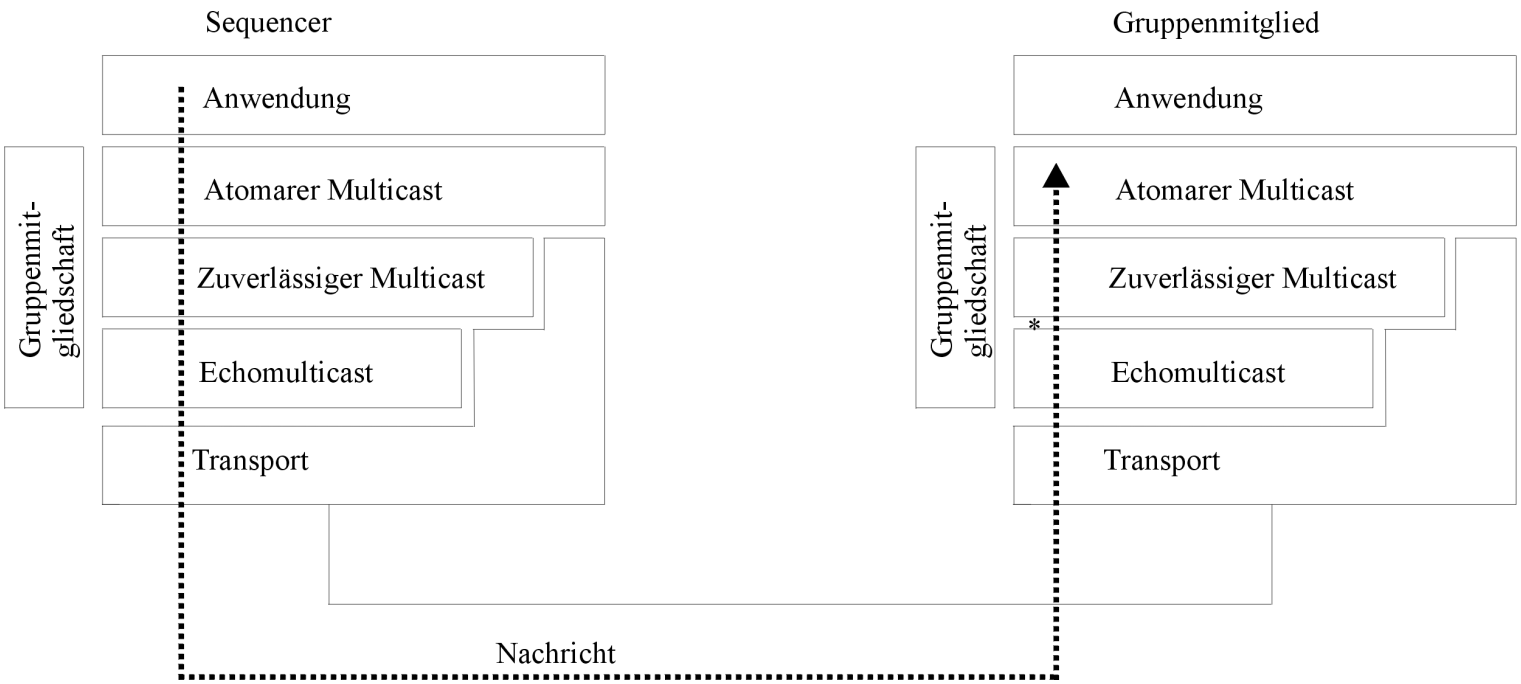
Atomarer Multicast - Sequencer

- Ausgewähltes Gruppenmitglied: Sequencer
- Sequencer legt Reihenfolge-Nachrichten fest

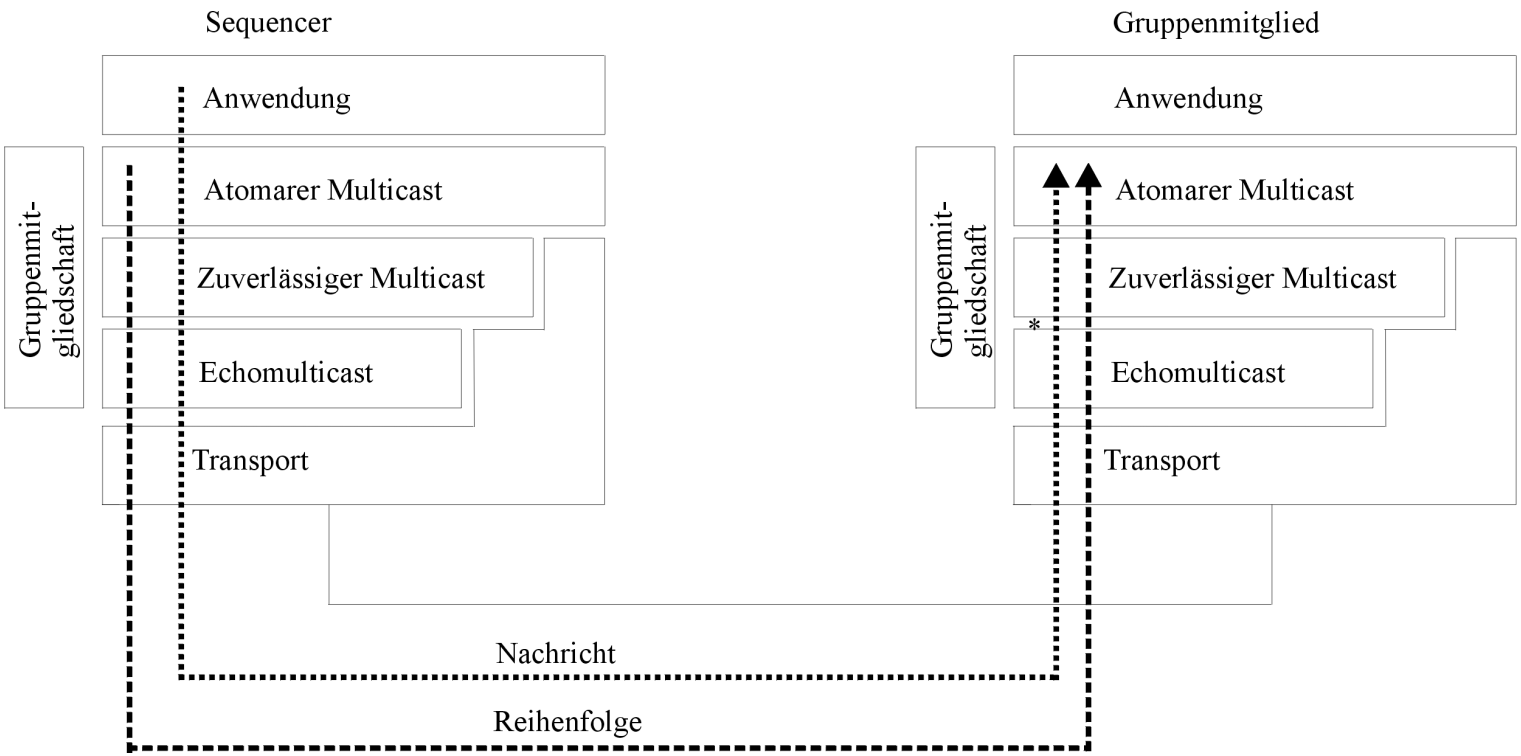
Atomarer Multicast - Prinzipieller Ablauf



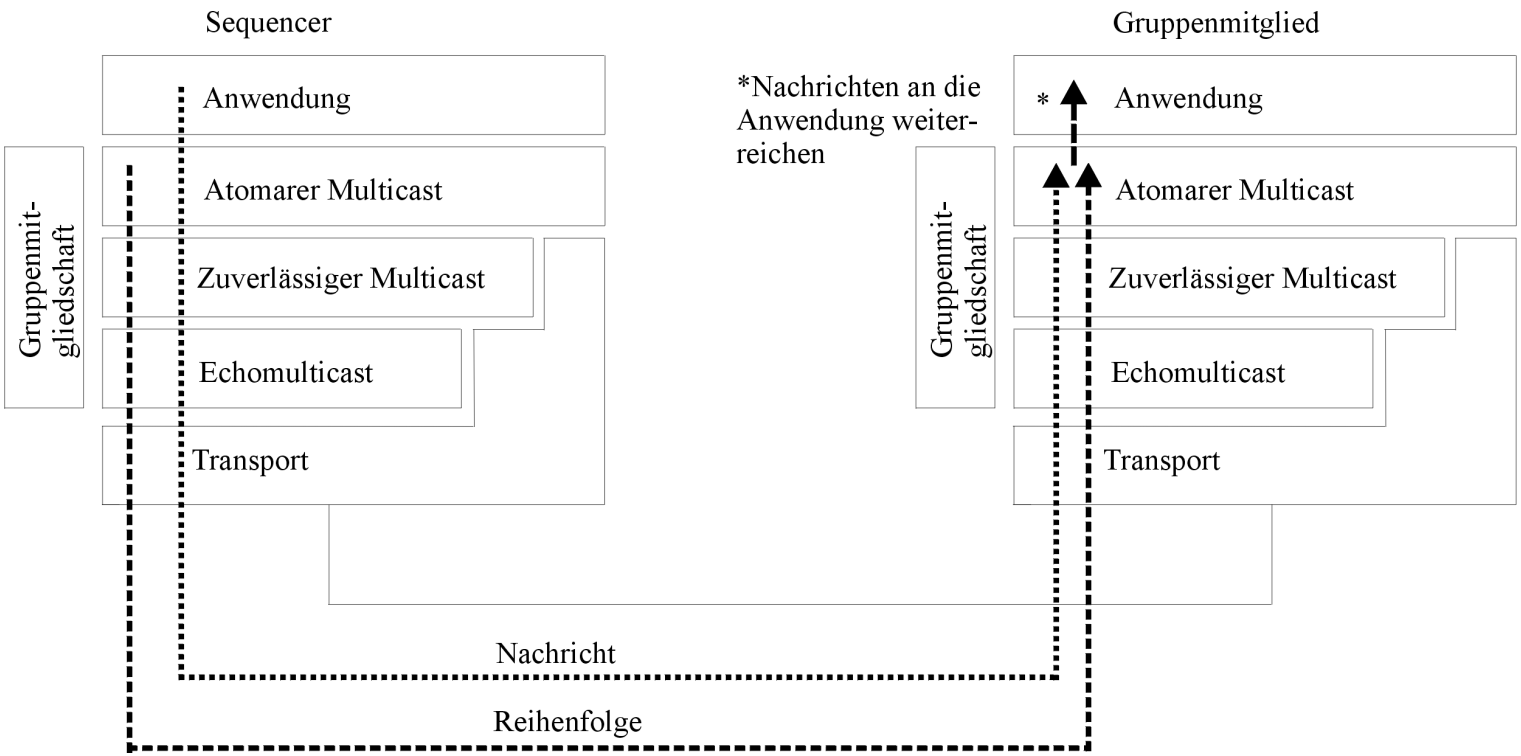
Atomarer Multicast - Prinzipieller Ablauf



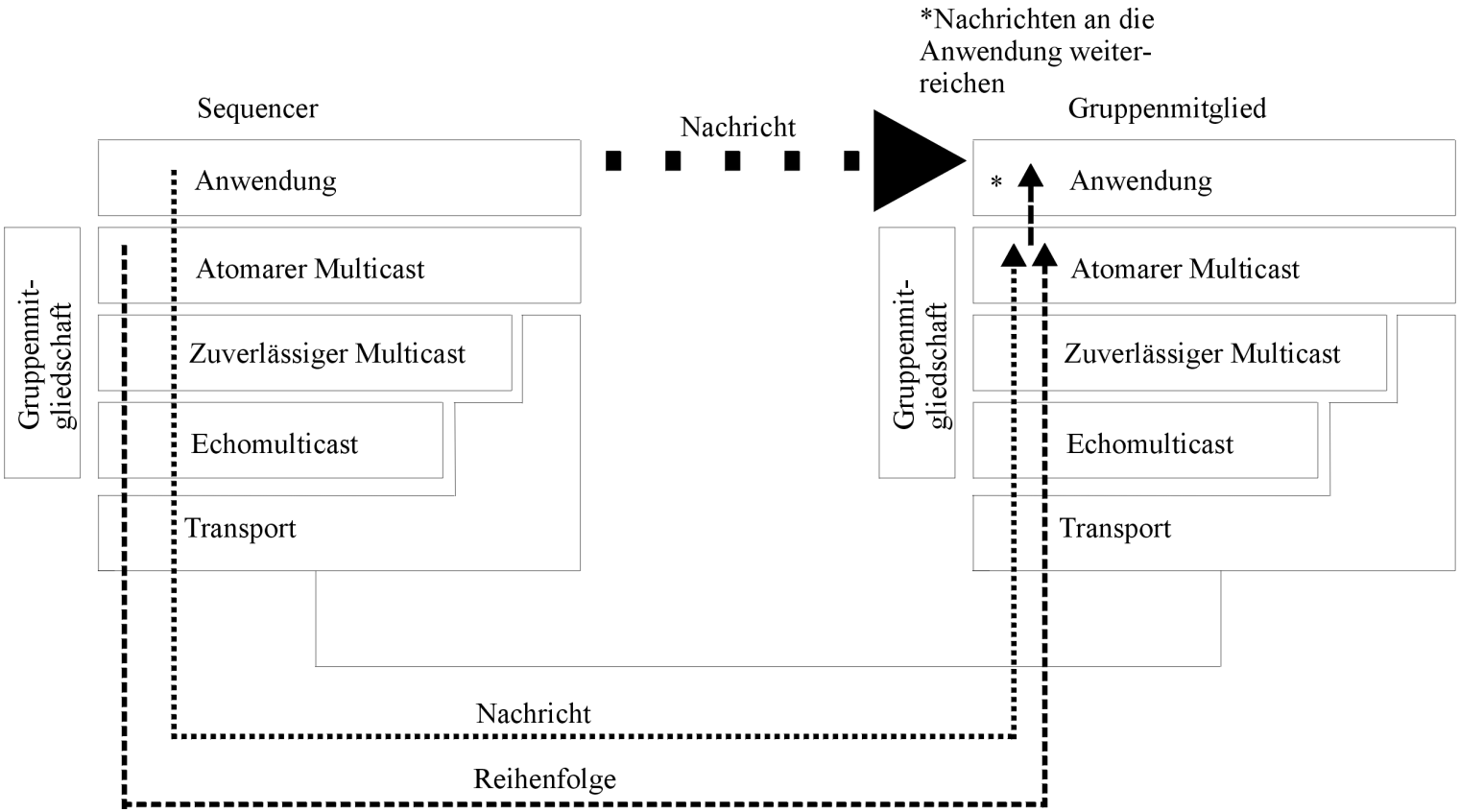
Atomarer Multicast - Prinzipieller Ablauf



Atomarer Multicast - Prinzipieller Ablauf



Atomarer Multicast - Prinzipieller Ablauf



Atomarer Multicast - Ablauf, Sonderfall

- Situation:
  - Neue Sicht wird geliefert
  - Ungelieferte Nachrichten der alten Sicht vorhanden
    - Erst Lieferung der Nachrichten der alten Sicht in deterministischer Reihenfolge, dann Lieferung der neuen Sicht

Atomarer Multicast - Sicherheit, Problemstellung

- Situation:
  Sequencer verhindert Weitergabe der Nachrichten an Anwendung
- Mögliche Wege:
  - Reihenfolge-Nachricht wird nie gesendet
  - Sequencer ordnet Senden einer nicht existenten Nachricht an; andere Nachrichten müssen warten

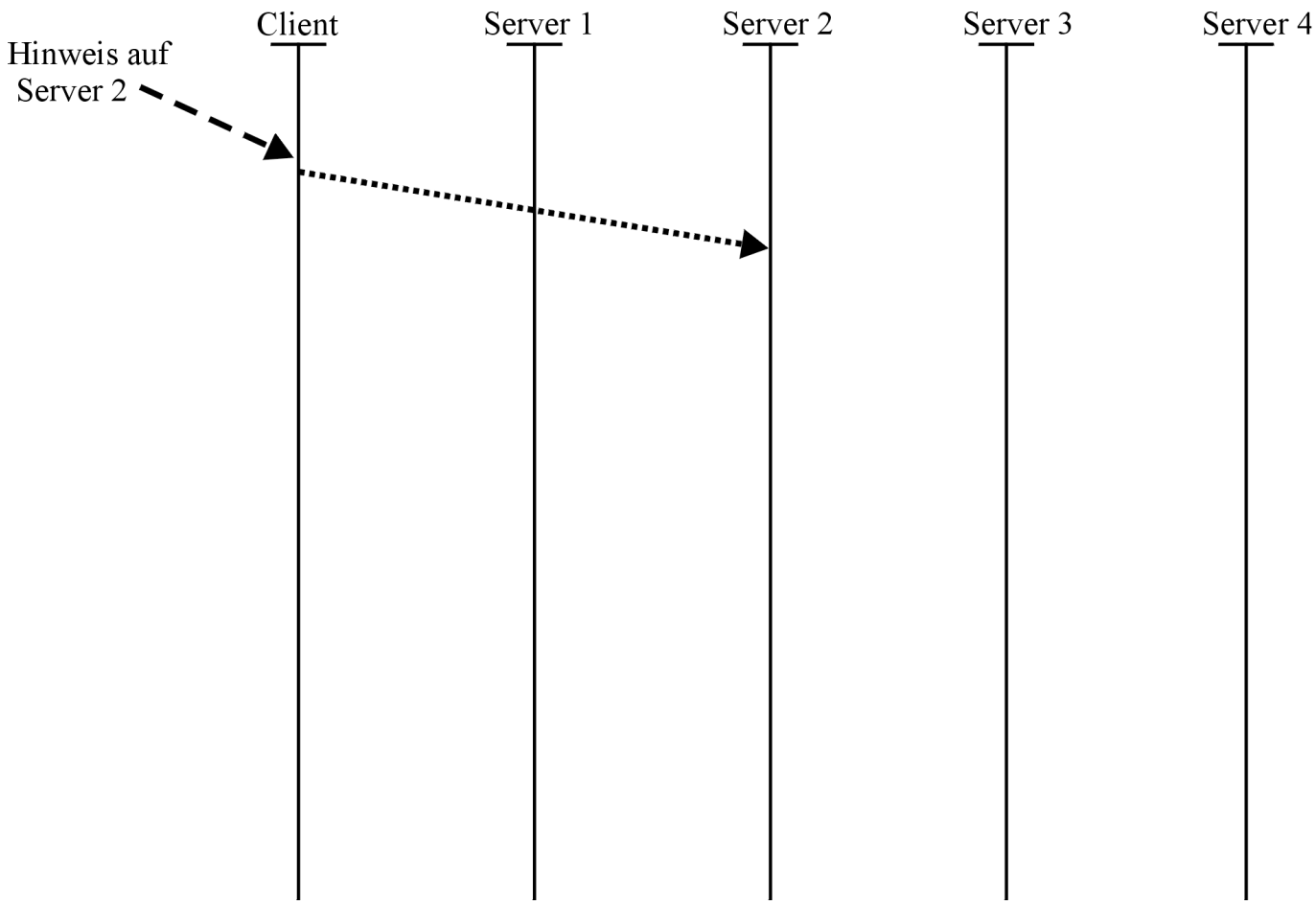
Atomarer Multicast - Sicherheit, Lösung

- Entfernung des Sequencers nach Timeout, wenn Nachricht erfolgreich an Prozess ausgeliefert und Reihenfolge-Nachricht fehlt

Clients - Ablauf

- Client muss nur einen Server lokalisieren können:
  - Broadcast
  - (Externer) Hinweis
- Vorgehen:
  1. Client schickt Anfrage an beliebigen Server

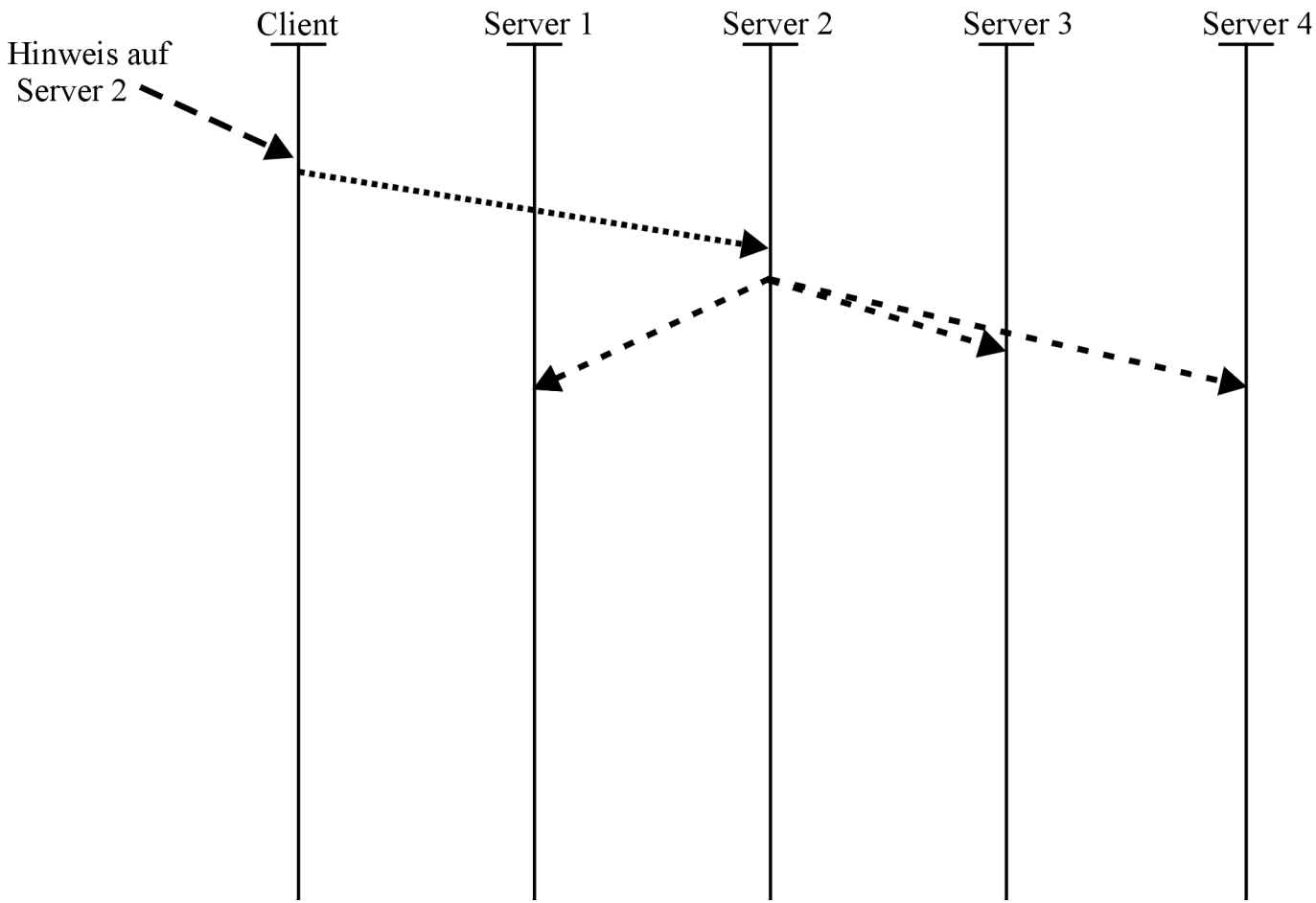
Clients - Ablauf



Clients - Ablauf

- Vorgehen:
  1. Client schickt Anfrage an beliebigen Server
  2. Server schickt Anfrage mit atomarem Multicast an Servergruppe

Clients - Ablauf



Clients - Bewertung

- Vorteile:
  - Client muss nur einen Server lokalisieren können
  - Server sammelt Anfragen von Clients und schickt nur ein Paket

Clients - Bewertung

- Nachteile:
  - Client schickt Anfrage an Server, Server arbeitet fehlerhaft; Manipulation oder Blockieren der Anfrage
  - Beide Fälle können erkannt werden

Output-Voting

- Zwei Output-Voting Protokolle:
  1. Output-Voting bei Clients
  2. Output-Voting bei Servern

Output-Voting - Bewertung

- Output-Voting bei Clients
- Nachteil:
  - Client muss jeden Server selbst identifizieren

Output-Voting - Bewertung

- Output-Voting bei Servern
- Vorteil:
  - Client muss *nicht* jeden Server selbst identifizieren, sondern nur den Dienst
- Nachteile:
  - Schlüssel des Dienstes muss unter den Servern verteilt werden (Aufwand!)

Output-Voting - Bewertung

- Evtl. Spezialhardware für RSA notwendig

Zusammenfassung

- State Machine Replication
- Atomarer Multicast
- Gruppenmitgliedschaft
- Echomulticast
- Zuverlässiger Multicast
- Atomarer Multicast bei Servern

Zusammenfassung

- Atomarer Multicast bei Clients
- Output-Voting