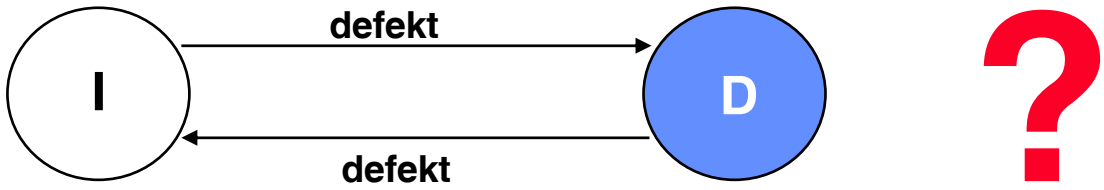
Wie Testen ?

Systemdiagnose zur Fehlererkennung



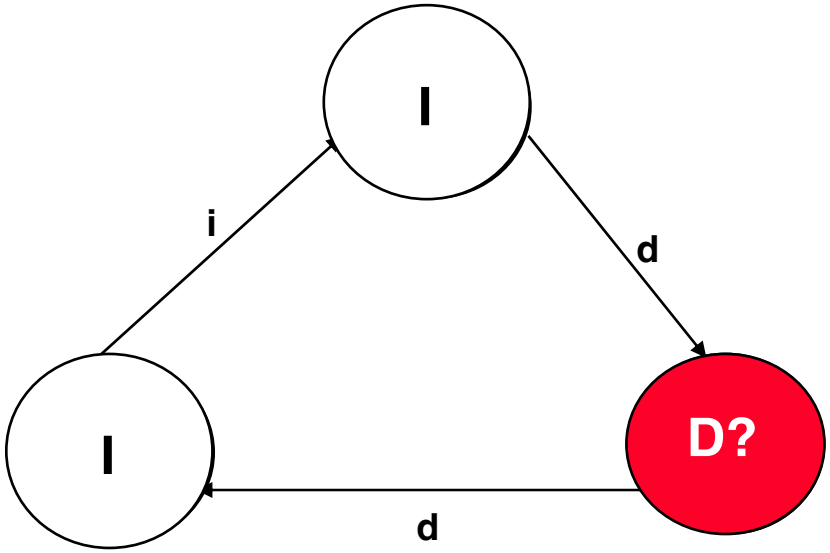
Annahmen:

- **Komponenten sind entweder korrekt oder defekt.**
- **ein Test ist vollständig und korrekt.**
- **ein korrekter Prozeß liefert ein korrektes Ergebnis.**
- **ein defekter Prozeß liefert ein ein beliebiges Ergebnis.**
- **ein zentraler (korrekter) Beobachter wertet den Test aus.**

F. P. Preparata, G. Metze, and R. T. Chien. On the connection assignment problem of diagnosable systems. IEEE Trans. Electron. Comput., EC--16:848--854, 1967

f–Diagnostizierbarkeit

1-diagnostizierbares System



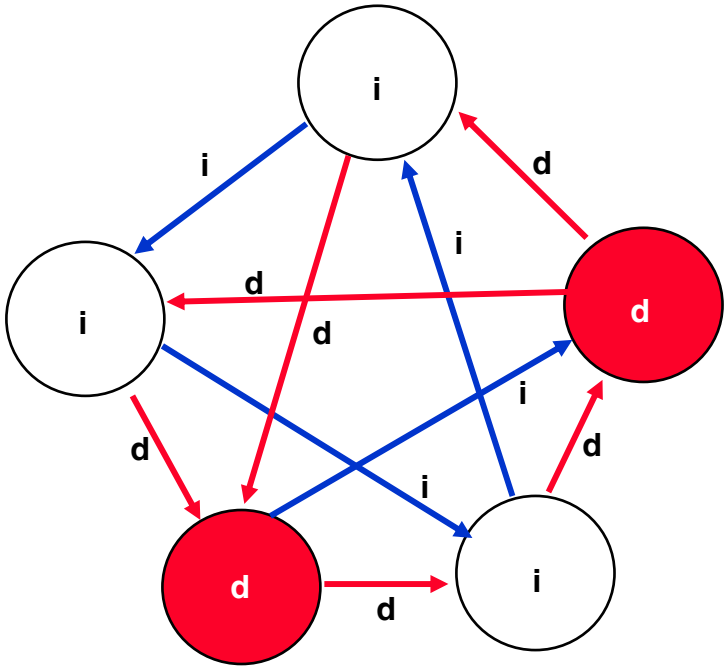
Annahmen:

- **Komponenten sind entweder korrekt oder defekt.**
- **ein Test ist vollständig und korrekt.**
- **ein korrekter Prozeß liefert ein korrektes Ergebnis.**
- **ein defekter Prozeß liefert ein ein beliebiges Ergebnis.**
- **ein Knoten wird als „defekt“ markiert, wenn er eine eingehende Kante von einem intakten Knoten hat, der ihn als „defekt“ getestet hat.**
- **ein zentraler Beobachter wertet den Test aus.**

f–diagnostizierbar:

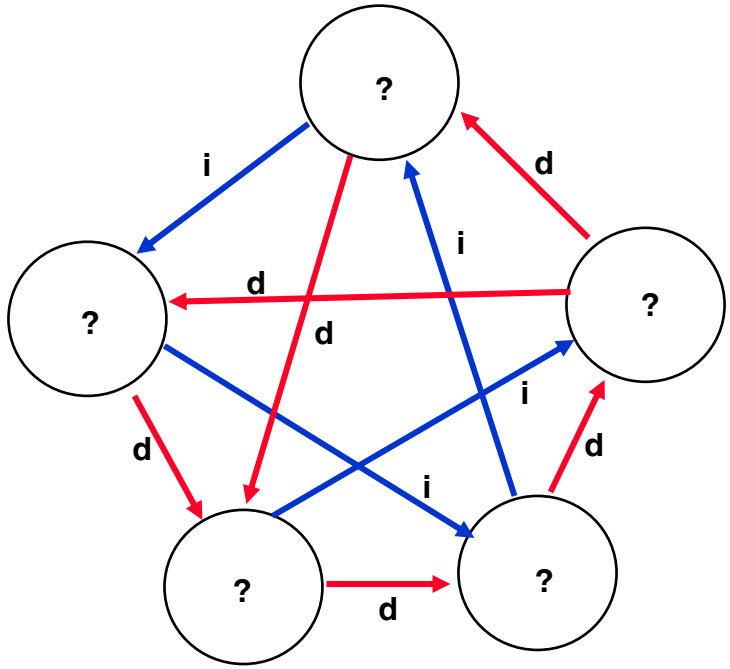
Ein System mit n Komponenten ist f–diagnostizierbar wenn $n \geq 2f + 1$ und jede Komponente mindestens f Komponenten testet, wobei sich die Komponenten nicht gegenseitig testen.

2-diagnostizierbares System



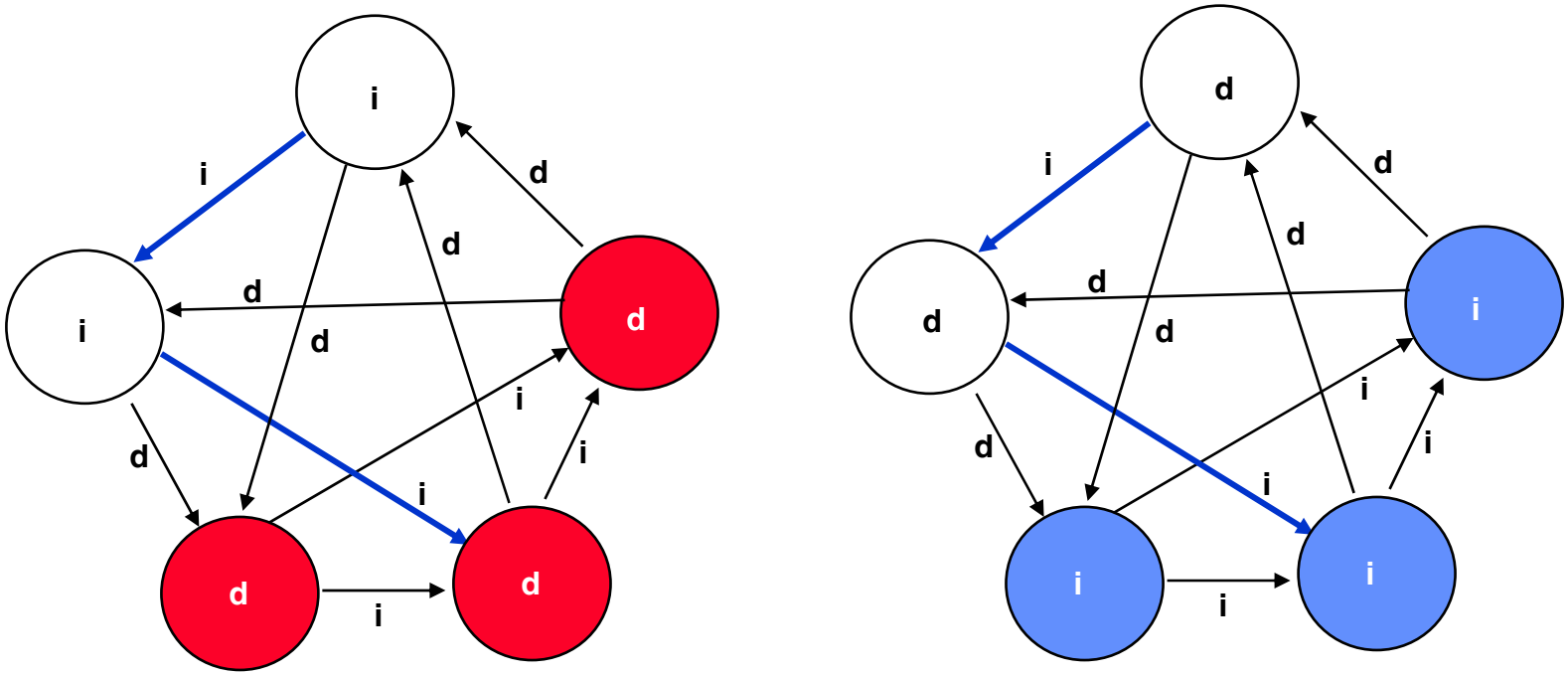
Annahmen:

- **Komponenten sind entweder korrekt oder defekt.**
- **ein Test ist vollständig und korrekt.**
- **ein korrekter Prozeß liefert ein korrektes Ergebnis.**
- **ein defekter Prozeß liefert ein ein beliebiges Ergebnis.**
- **ein Knoten wird als „defekt“ markiert, wenn er eine eingehende Kante von einem intakten Knoten hat, der ihn als „defekt“ getestet hat.**
- **ein zentraler Beobachter wertet den Test aus.**



Gibt es ein eindeutiges Ergebnis der Diagnose?

3 defekte Knoten



Fehler kann (natürlich) nicht erkannt werden,
weil er die Fehlerannahme (max 2 Fehler) verletzt.

Fehlererkennung in verteilten Systemen

System-Modell: kooperierende Prozesse, die über Nachrichten kommunizieren.

Fehlermodell 1 (Crash-F-Sematik):
Prozesse können abstürzen, die Kommunikation ist zuverlässig.

Fehlermodell 2 (Omission-F-Semantik):
Prozesse können abstürzen und Nachrichten können ausbleiben.

Fehlermodell 3 (Performance-F-Semantik):
Prozesse können abstürzen, Nachrichten können ausbleiben und verspätet ankommen.

Fehlermodell 4 (Byzantinische-F-Semantik):
Prozesse können beliebige Fehler aufweisen.

Fehlerdetektoren und Konsistenz verteilter Fehlererkennung

Intuitives Konsistenzkriterium:
Wenn ein Prozeß ausfällt, erkennen alle intakten Prozesse diesen Ausfall und erreichen Konsens über fehlerhafte Prozesse.

Formalisierung (Chandra, Tueg 1996):

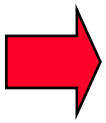
Strenge Konsistenz (SK): **Ein korrekter Prozeß wird nie als ausgefallen erkannt. (Sicherheitskriterium)**

Strenge Vollständigkeit (SV): **Ein Ausfall wird (irgendwann) von jedem korrekten Prozeß erkannt (Lebendigkeitskriterium)**

Unter welchen Bedingungen können SK und SV erreicht werden ?

Annahmen:

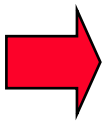
1. **Die Laufzeit der Nachrichten ist beschränkt,**
2. **Die Prozesse können ein Lebenszeichen in einem beschränkten Zeitintervall erzeugen.**
3. **Fehlermodell 1**



„Herzschlag“ – Mechanismus ist perfekter Fehlerdetektor

Annahmen:

1. **Die Laufzeit der Nachrichten ist beschränkt,**
2. **Die Prozesse können ein Lebenszeichen in einem beschränkten Zeitintervall erzeugen.**
3. **Fehlermodell 2, wobei die Anzahl der Omissions beschränkt ist.**

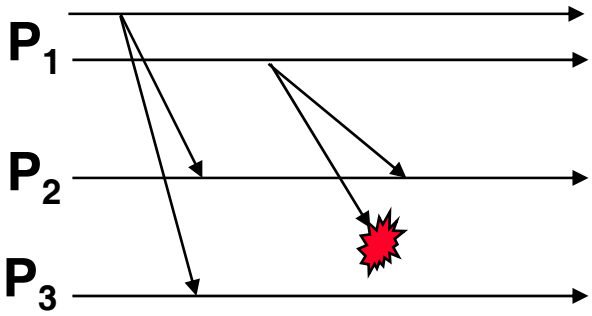


Einsatz von Mechanismen zur Maskierung von Omissions

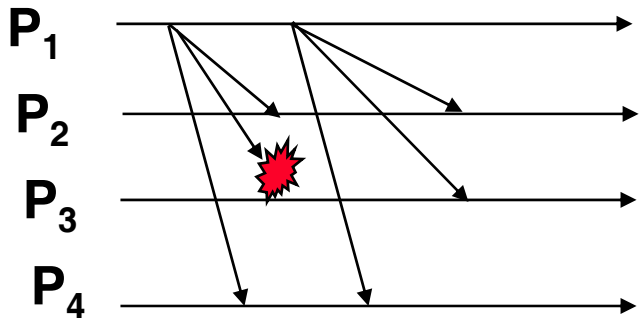
FT Kommunikation - Behandlung von Nachrichtenfehlern:

Statische Redundanz: Maskierend

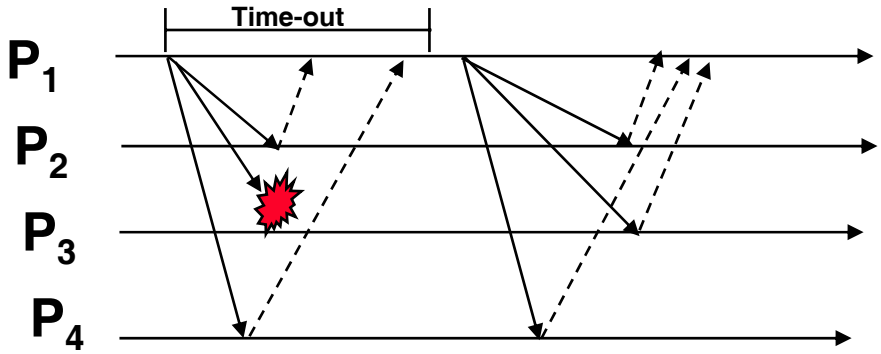
Komponentenredundanz



Zeitredundanz

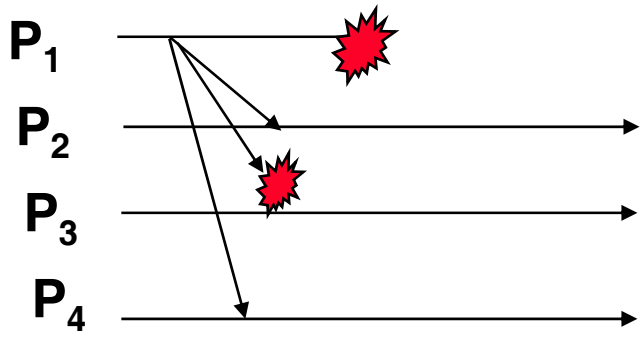


Dynamische Redundanz: Erkennung, Recovery

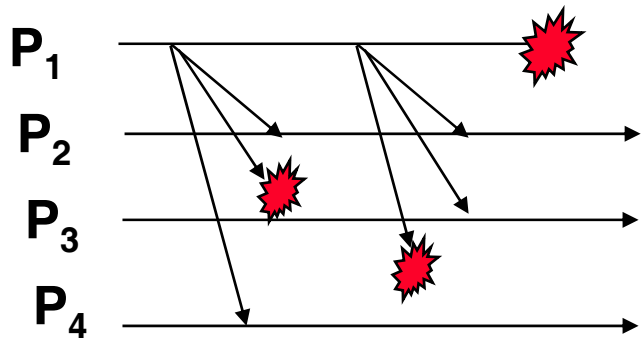


FT Kommunikation - Behandlung von Senderfehlern:

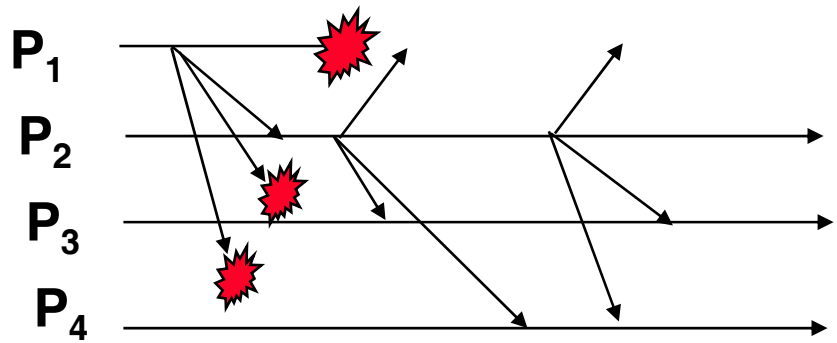
Unzuverlässiger Multicast



Best effort Multicast



Zuverlässiger Multicast



Nicht perfekte Fehlererkennung

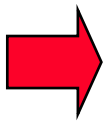
Annahmen:

Zeitliche:

1. **Die Laufzeit der Nachrichten ist nicht beschränkt,**
2. **Die Prozesse können ein Lebenszeichen nicht in einem beschränkten Zeitintervall erzeugen.**

Anzahl der Fehler:

3. **Die Anzahl der Omissions kann nicht beschränkt werden.**



Entscheidung, ob ein Prozeß ausgefallen ist oder nicht, ist nicht deterministisch möglich.

Konsensbildung in verteilten Systemen

Eine Gruppe von Prozessen einigt sich auf einen gemeinsamen Wert.

Dabei müssen folgende Eigenschaften erfüllt sein:

Konsistenz: **Alle Prozesse einigen sich auf denselben Wert und die Entscheidung ist endgültig.**

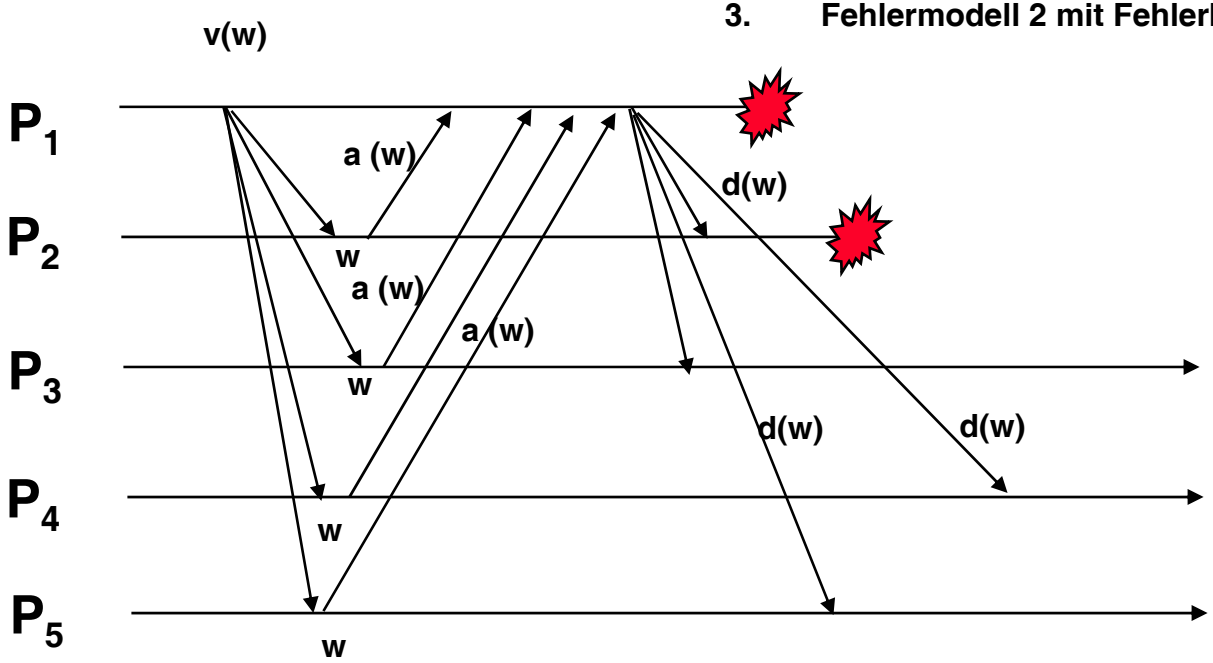
Nicht-Trivialität: **Der Wert, auf den sich geeinigt wurde muß der Eingabewert eines Prozesses sein (oder eine Funktion dieses Eingabewertes).**

Terminierung: **Jeder korrekte Prozeß entscheidet auf einen gemeinsamen Wert innerhalb eines endlichen Zeitintervalls.**

Fehlertolerante Konsensbildung

Annahmen:

1. **Die Laufzeit der Nachrichten ist beschränkt,**
2. **Die Fehlererkennung ist zuverlässig.**
3. **Fehlermodell 2 mit Fehlerbehandlung**

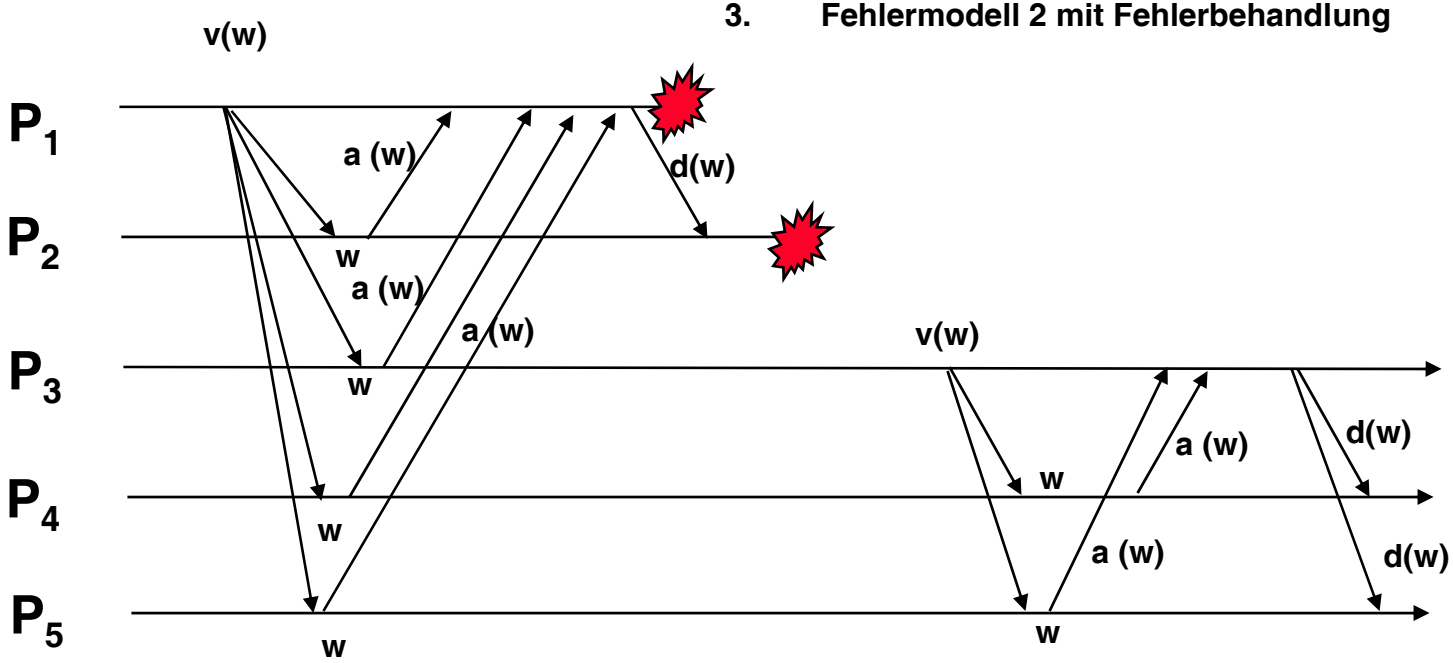


v(w): vorschlagen(w)
a(w): akzeptiert (w)
d(w): entschieden (w)

Fehlertolerante Konsensbildung

Annahmen:

1. **Die Laufzeit der Nachrichten ist beschränkt,**
2. **Die Fehlererkennung ist zuverlässig.**
3. **Fehlermodell 2 mit Fehlerbehandlung**

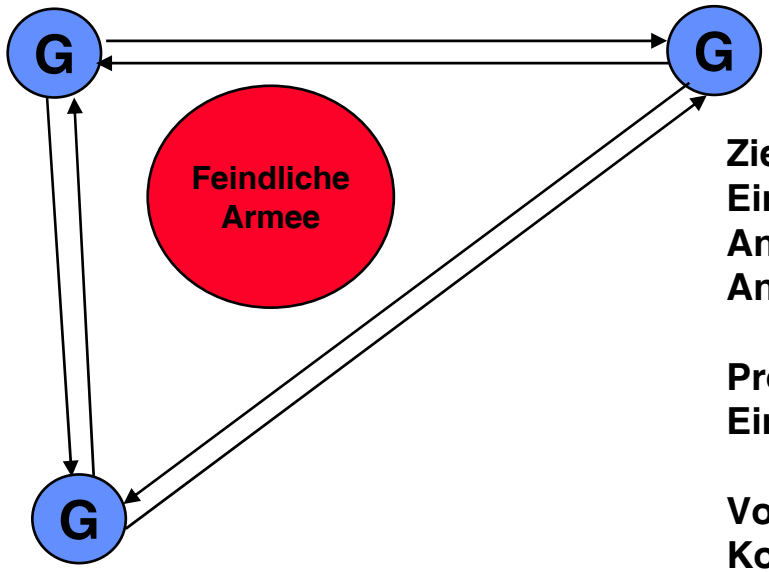


v(w): vorschlagen(w)
a(w): akzeptiert (w)
d(w): entschieden (w)

Byzantinische Fehler und Byzatinische Einigung

L. Lamport, R. Shostak, M. Pease: „The byzantine generals‘ problem“, ACM TC on Progr. Languages and systems, 4(3), 1982

Die Story:

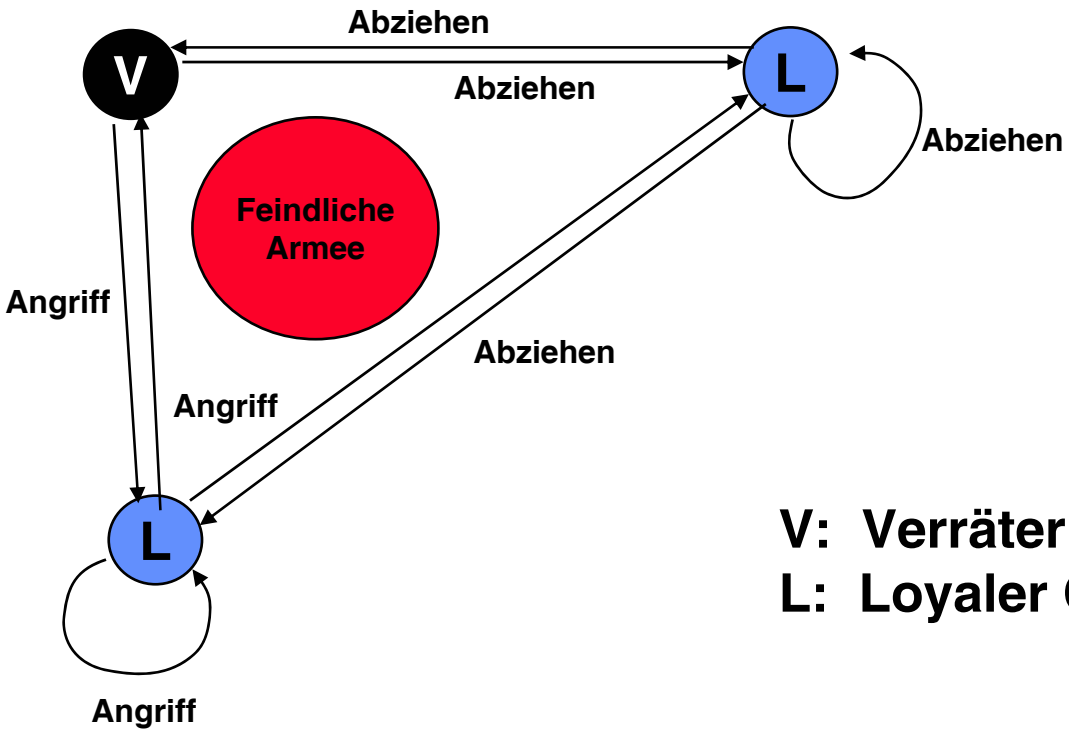


Ziel:
Einigung auf einen gemeinsamen Plan, Angreifen oder Abziehen? Nur bei einem gemeinsamen Angriff kann man erfolgreich sein.

Problem:
Ein möglicher Verräter

Voraussetzung
Kommunikation nur über „reitende Boten“ (Punkt-zu-Punkt-Nachrichten). Die sind allerdings zuverlässig!

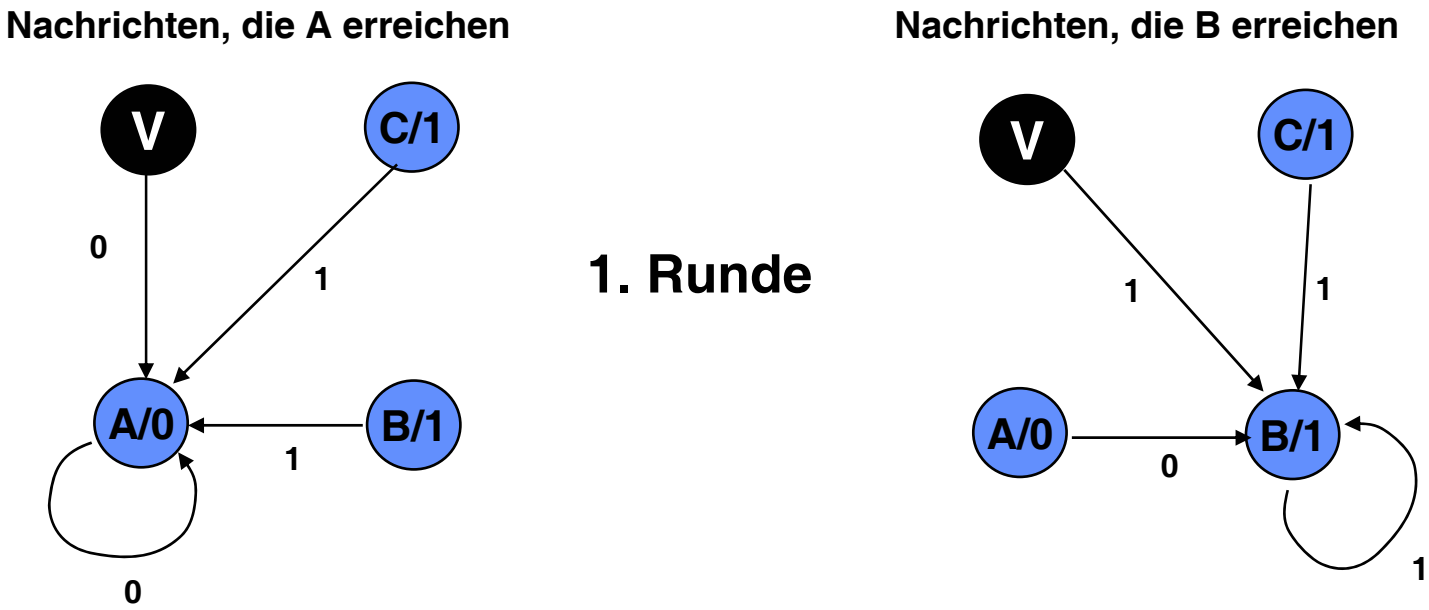
Unter welchen Umständen und mit welchem Protokoll kann eine vertrauenswürdige Mehrheitsentscheidung herbeigeführt werden?



V: Verräter
L: Loyaler General

Auch mehrere Runden der Einigungsversuche helfen nicht, weil ein loyaler General nicht weiß, wer der Verräter ist.

Einigung auf einen Wert in zwei Runden:

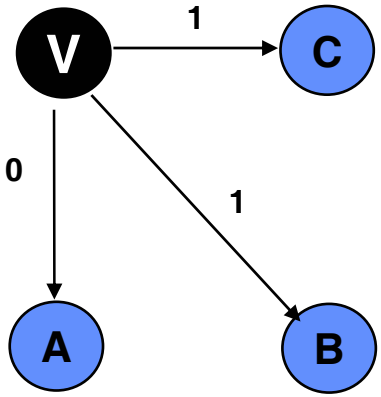


Verteilung der Werte

In der ersten Runde kann noch keine eindeutige Entscheidung getroffen werden, da A und B nicht übereinstimmen.

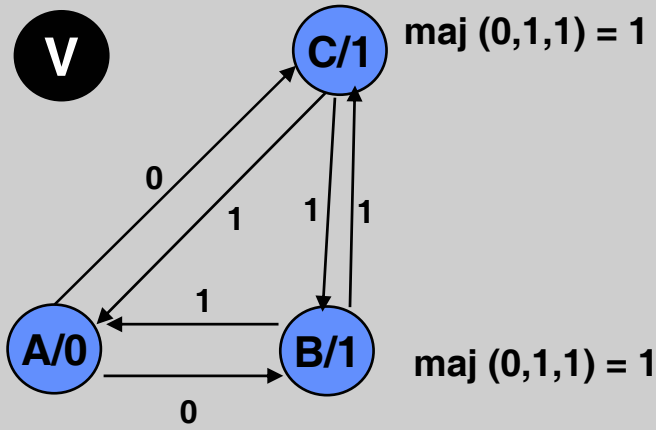
1. Runde

Verteilen der Werte von einem Teilnehmer



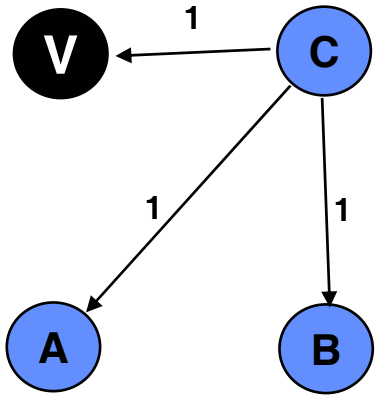
2. Runde

Einigung auf einen Wert, den ein Teilnehmer geschickt hat.



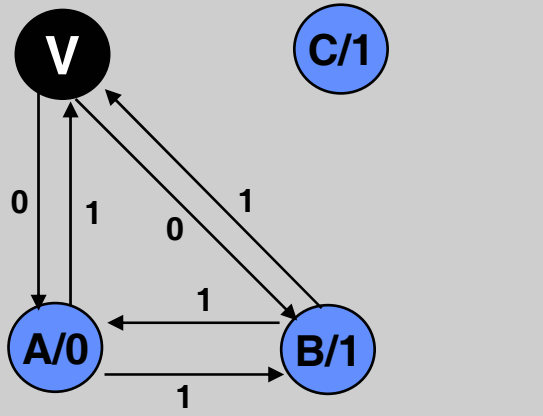
1. Fall

Sender ist Verräter



2. Fall

Verräter verfälscht bei Weiterleitung



Lokale Entscheidung für einen Wert

- Als Teilnehmer werden Prozesse angenommen.

- Jeder Prozeß entscheidet lokal durch Majoritätsvotum den Wert, den jeder andere Knoten einnimmt.

- Der Wert, der von einer Mehrheit der Prozesse gewählt wurde, gilt als gemeinsamer Wert.

- Zur Erkennung von f byzantinischen Fehler werden mindestens

$(3f + 1)$ Prozesse benötigt.