Grundlagen zuverlässiger fehlertoleranter Systeme

Quellen:

Eugen Schäfer: "Zuverlässigkeit, Verfügbarkeit und Sicherheit in der Elektronik, Eine Brücke von der Zuverlässigkeittheorie zu den Aufgaben der Zuverlässigkeits-praxis", 1. Auflage, Vogel Verlag, 1979, ISBN 3-0823-0586-8,

Karl-Erwin Großpietsch: "Zuverlässigkeitstheoretische Grundlagen", GMD-Seminar, St. Augustin

Stefan Poledna: "Lecture on Fault-Tolerant Systems", Vorlesungsfolien, Institut für Technische Informatik, TU Wien, SoSe 1996

Grundlagen zuverlässiger Systeme

Verläßlichkeit (Zuverlässigkeit):

Die Verläßlichkeit (Dependability) eines Systems ist die Qualität einer vom System erbrachten Funktion (Service), in die begründbar und berechtigterweise Vertrauen (reliance) gesetzt werden kann.

Die Funktion (Service) ist das an der Schnittstelle zu anderen Systemen, die mit dem betrachteten System interagieren, beobachtbare Systemverhalten. Die Qualität bezieht sich auf die Übereinstimmung der erbrachten mit der spezifizierten Systemfunktion.

Terminologie:

- **Beeinträchtigungen (Impairments) : Fehler**
- **Attribute**
- **Maße**
- **Mechanismen**

Laprie, J.-C. : Dependability: A unifying concept for reliable, safe, secure computing.
In IFIP Congress, volume 1, (1992)pages 585-593.

Attribute der Zuverlässigkeit (Dependability)

Überlebensfähigkeit (Reliability) bedeutet Zuverlässigkeit in Hinblick auf ununterbrochenes korrektes Systemverhalten. Es ist als die Wahrscheinlichkeit definiert, daß ein zu Beginn fehlerfreies System bis zu einem bestimmten Zeitpunkt fehlerfrei bleibt.

Verfügbarkeit (Availability) bedeutet Zuverlässigkeit in Hinblick auf die momentane Bereitschaft eines Systems zur Erbringung eines Service. Verfügbarkeit wird als quantitaves Maß definiert, das die Ausfalldauer zu der Dauer korrekten Systemverhaltens in Beziehung setzt, d.h. die Wahrscheinlichkeit, das System zu einem beliebigen Zeitpunkt fehlerfrei anzutreffen.

Prozeßsicherheit (Safety) bedeutet Zuverlässigkeit in Hinblick auf das Verhindern katastrophaler Auswirkungen eines Systemverhaltens auf seine Umgebung, wobei mit Umgebung meist die physikalische, reale Umgebung gemeint ist, wie z.B. industrielle Prozeßsteuerungsanlagen, Kraftwerke, Verkehrslenkungssysteme, u.s.w.

Informationssicherheit (Security) bedeutet Zuverlässigkeit in Hinblick auf die Erhaltung der Vertraulichkeit (Confidentiality) und Integrität (Integrity) von Information in einem Computersystem.

Quantitative Ermittlung der Zuverlässigkeit

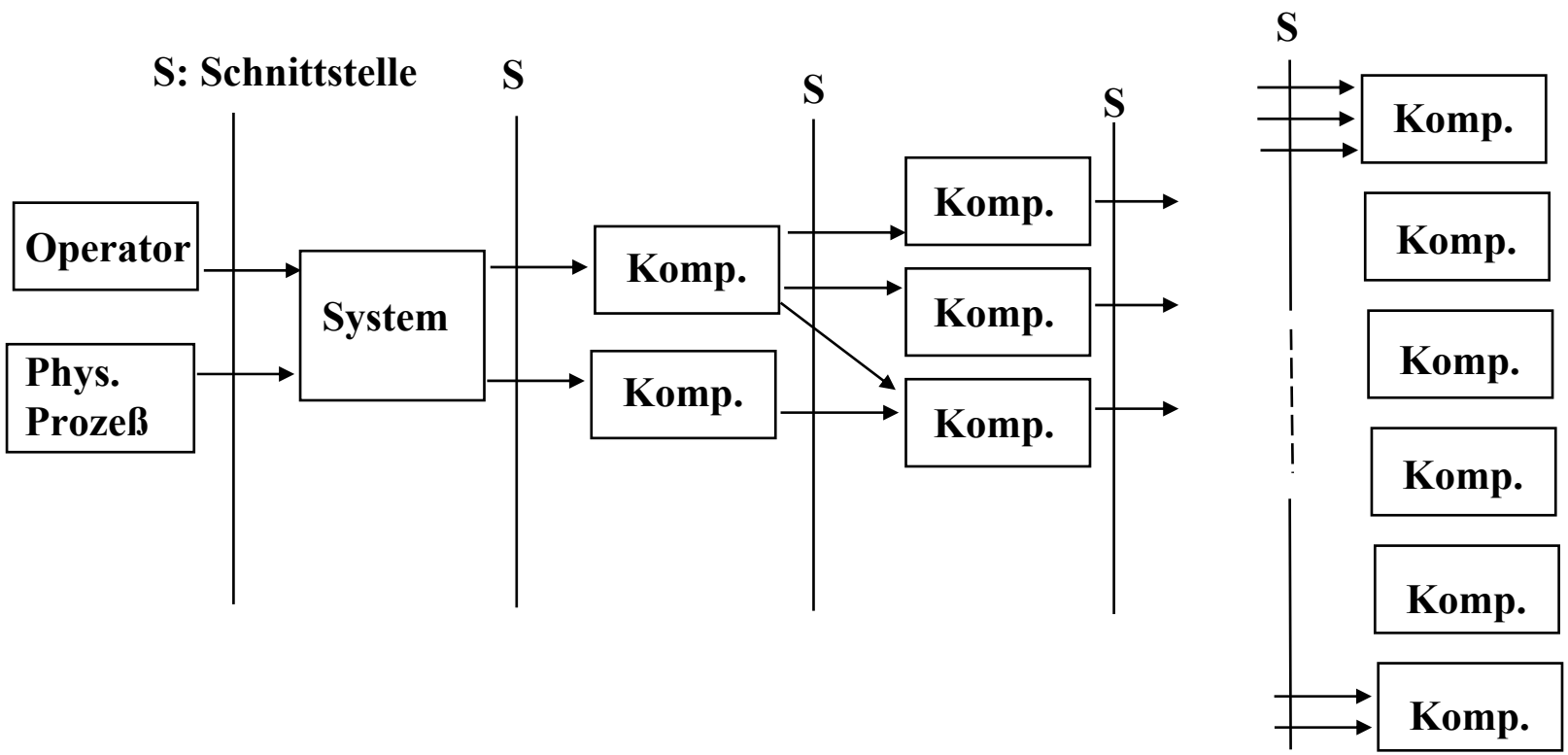
Strukturbasierte Modellierung:

- **identifizierbare unabhängige Komponenten**
- **jede Komponente besitzt eine bestimmte Zuverlässigkeit**
- **die Konstruktion des Modells basiert auf der Verbindungsstruktur zwischen den Komponenten**

Ein System wird definiert durch:

- **seine Struktur, d.h. die Topologie seiner Komponenten**
- **sein Verhalten, d.h. durch die Gesamtheit des Verhaltens seiner Komponenten**

Systemkomponenten sind hierarchisch organisiert. Dadurch ergibt sich eine Abhängigkeitsrelation (→) zwischen den Systemebenen.



Quantitative Ermittlung der Zuverlässigkeit durch Zuverlässgkeitsschaltbilder

Intaktwahrscheinlichkeiten:

Für jeden Teil eines Systems werden zwei Zustände betrachtet:

• intakt (funktionstüchtig)
• defekt (ausgefallen)

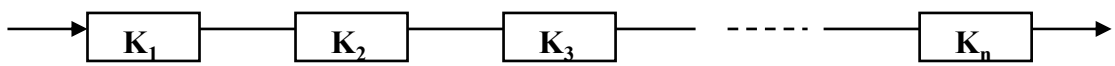
Intakt-Wahrscheinlichkeit einer Komponente oder eines Systems von Komponenten:
Wahrscheinlichkeit, dass die Komponente oder das System das spezifizierte Verhalten zeigen.

Ein System ist fehlertolerant, wenn es intakt sein kann, ohne dass alle Komponneten intakt sind.

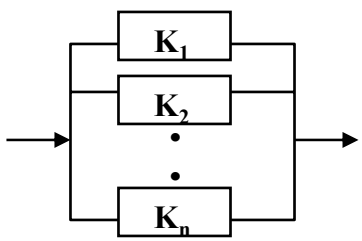
Zuverlässigkeits-Schaltbilder (nicht zu verwechseln mit elektrischen Schaltbildern) :

Abstraktion eines Systems in Komponenten, denen jeweils eine spezifische Zuverlässigkeit zugeordnet wird.

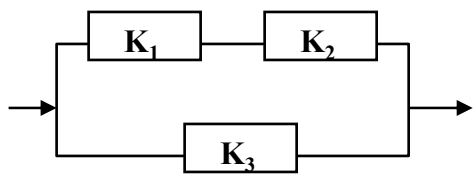
• **Seriensystem:**



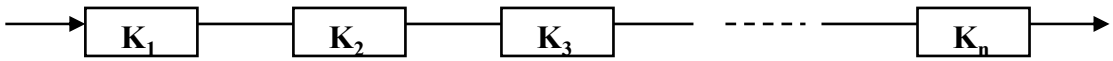
• **Parallelsystem:**



• **Serien/Parallelsystem:**



Intaktwahrscheinlichkeit für ein Seriensystem:



P_{serie} = P (K_1 intakt) und P(K_2 intakt) und .....P(K_n intakt)

Annahme: Die Eigenschaften (K_i intakt) (i=1,..,n) sind unabhängig.

⇒ P_{serie} = P (K_1 intakt) • P(K_2 intakt) • ..... •P(K_n intakt)

mit p_i : Intaktwahrscheinlichkeit der Komponente i:

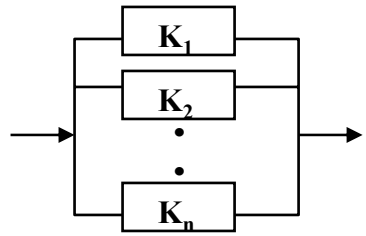
⇒ $P_{serie} = p_1 \bullet p_2 \bullet ..... \bullet p_n$

Beispiel:

n identische Komponenten:

$P_{serie} = p_i^n$, n = 5, p_i = 0,99: $P_{serie} = 0{,}99^5 = 0{,}95$
$P_{serie} = p_i^n$, n = 5, p_i = 0,70 : $P_{serie} = 0{,}70^5 = 0{,}16$

Intaktwahrscheinlichkeit für ein Parallelsystem (1-aus-n) :



Defektwahrscheinlichkeit = 1 - Intaktwahrscheinlichkeit
(Intakt und defekt sind zueinander komplementäre Ereignisse).

$P_{parellel}$ = P (K_1 defekt) und P(K_2 defekt) und .....P(K_n defekt)

Annahme: Die Eigenschaften (K_i defekt) (i=1,..,n) sind unabhängig.

⇒ $P_{parellel} = P(K_1 \text{ defekt}) \bullet P(K_2 \text{ defekt}) \bullet ..... \bullet P(K_n \text{ defekt})$

Mit p_i : Defektwahrscheinlichkeit der Komponente i:

⇒ $P_{parellel} = p_1 \bullet p_2 \bullet ..... \bullet p_n$

Beispiel Defektwahrscheinlichkeit:

n identische Komponenten:

$P_{serie} = p_i^n$, n = 5, p_i = 1 - 0,99: $P_{serie} = 0{,}01^5 = 0{,}0000000001$ Intaktw.: 0,9999999999
$P_{serie} = p_i^n$, n = 5, p_i = 1- 0,70 : $P_{serie} = 0{,}30^5 = 0{,}00243$ Intaktw.: 0,99757

K - aus - n - Systeme

Systeme aus n Komponenten von denen mindestens k der Komponenten intakt sind.

Wahrscheinlichkeit, daß genau k ausgewählte Komponenten intakt (die Komponenten 1,..,k), die anderen Komponenten defekt sind (die Komponenten k+1,...,n).

$$P_{k\text{-aus-}n} = p_1 \bullet p_2 \bullet .... \bullet p_k \bullet (1 - p_{k+1}) \bullet (1 - p_{i+2}) \bullet .... \bullet (1- p_n)$$

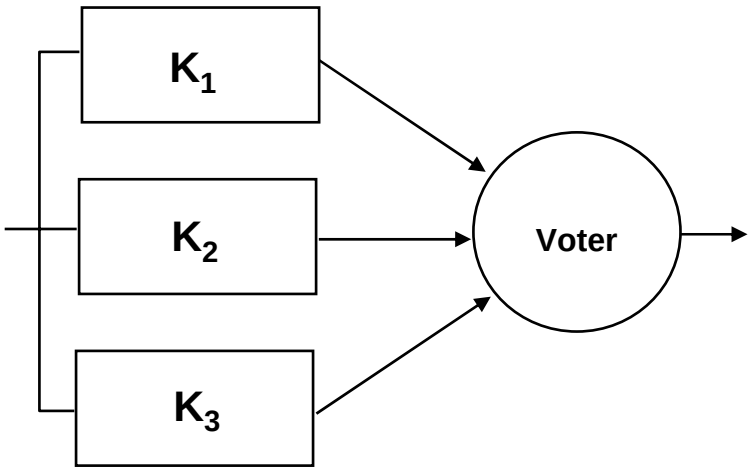
Es gibt $\binom{n}{i}$ **Möglichkeiten, i Komponenten aus n Komponenten auszuwählen:**

$$P_{k\text{-aus-}n} = \sum_{i=k}^{n} \binom{n}{i} \; p^i \bullet (1 - p)^{n- i}$$

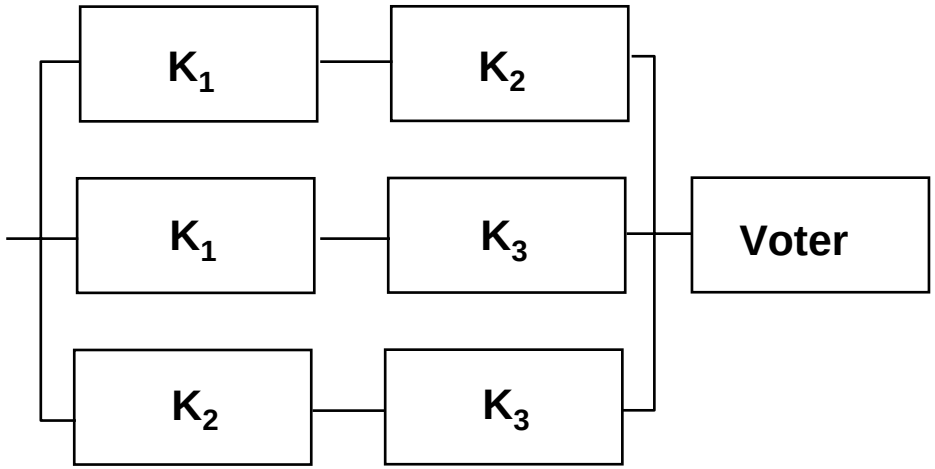
Beisp.: ein 2-aus-3 System: $\binom{3}{2} p^2 \bullet (1 - p)^{3-2} + \binom{3}{3} p^3 \bullet (1 - p)^{3-3} = 3 \bullet p^2 \bullet (1 - p) + p^3 \bullet 1$

Beispiel TMR (Triple Modulare Redundanz: 2-aus-3-System)

(Elektr.) Schaltbild



Zuverlässigkeits-Schaltbild



$$P_{TMR} = (p^3 + 3\,p^2 \cdot (1-p)\,) \cdot p_{voter}$$

$$p = 0{,}9,\ p_{voter} = 0{,}99:\ P_{TMR} = (0{,}9^3 + 3 \cdot 0{,}9^2 \cdot (1-0{,}9)) \cdot 0{,}99$$

$$= (0{,}729 + 2{,}43 \cdot 0{,}1) \cdot 0{,}99 = 0{,}972 \cdot 0{,}99$$

$$= 0{,}96228$$

Wie werden die Wahrscheinlichkeiten für eine intakte oder defekte Komponente über die Zeit ermittelt ?

Maße der Fehlertoleranz

Lebensdauer T

Zeit vom Beanspruchungsbeginn (DIN 40 042) bis zum Totalausfall (nicht mehr reparierbar)

Ausfallwahrscheinlichkeit F(t)

ist die Wahrscheinlichkeit für eine Komponente bis zum Zeitpunkt $T < t_i$ auszufallen.

Überlebenswahrscheinlichkeit R(t) (Reliability)

Wahrscheinlichkeit, daß eine Komponente zum Zeitpunkt t_i noch nicht ausgefallen ist. F(t) ist das Komplement zu R(t).

$$R(t) = 1 - F(t)$$

Für nicht reparierbare Systeme ist R(t) eine monoton fallende Funktion. $R(0) \leq 1$, $R(\infty) = 0$

Ausfallwahrscheinlichkeitsdichte f(t)

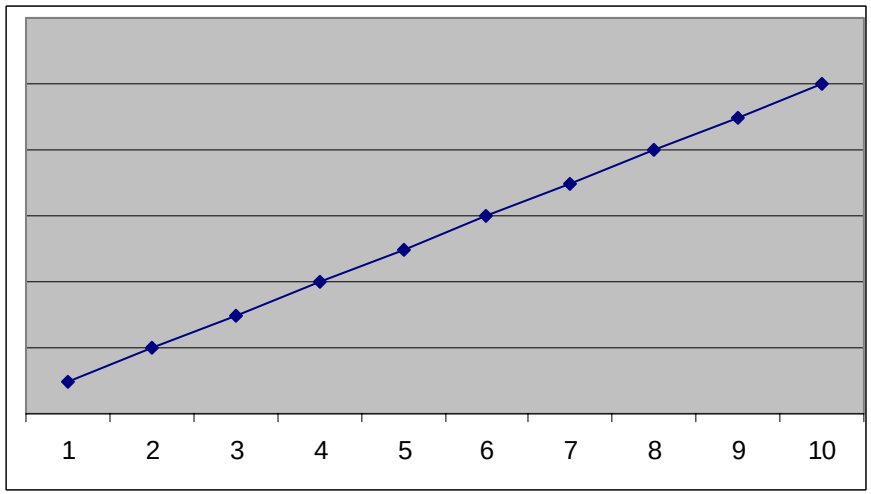
f(t) • dt ist die Wahrscheinlichkeit, daß der Ausfall einer Komponente im Zeitintervall (t, t+dt) auftritt.

f(t) ist dann die Wahrscheinlichkeit, mit der in diesem Zeitintervall Ausfälle erwartet werden können.

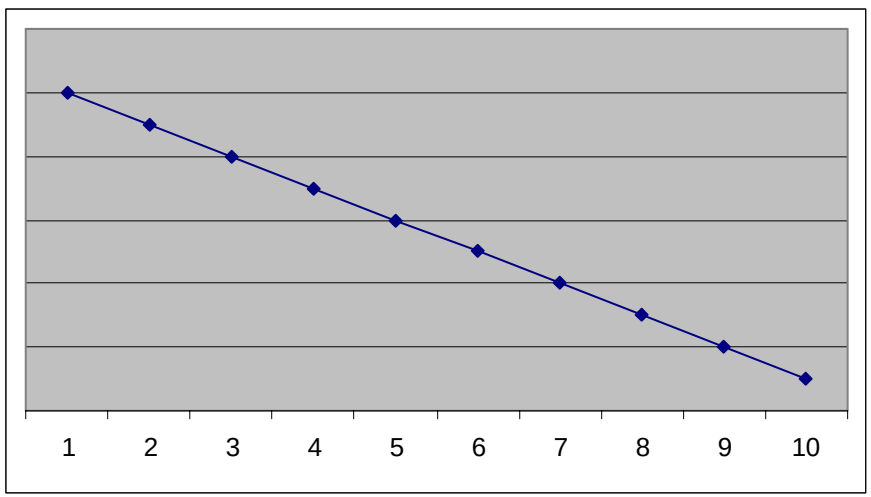
$$f(t) = \frac{dF(t)}{dt} = - \frac{dR(t)}{dt}$$

Konstante Ausfallwahrscheinlichkeitsdichte

F(t)

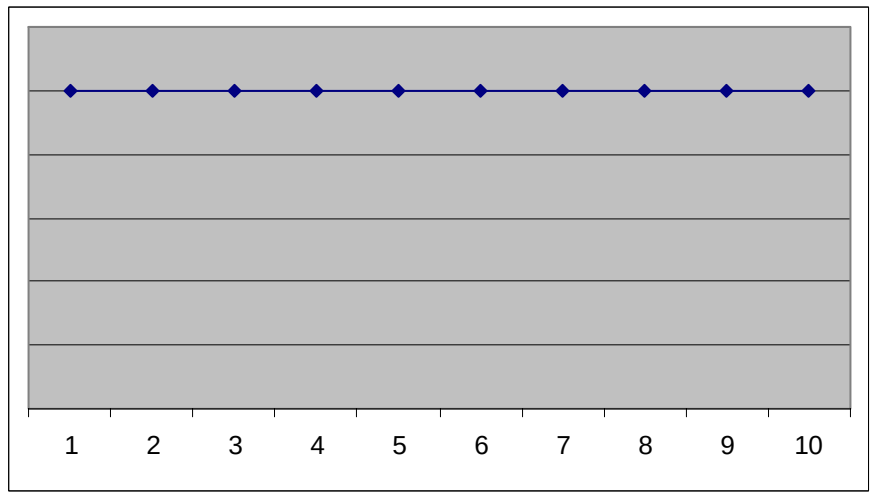


R(t)



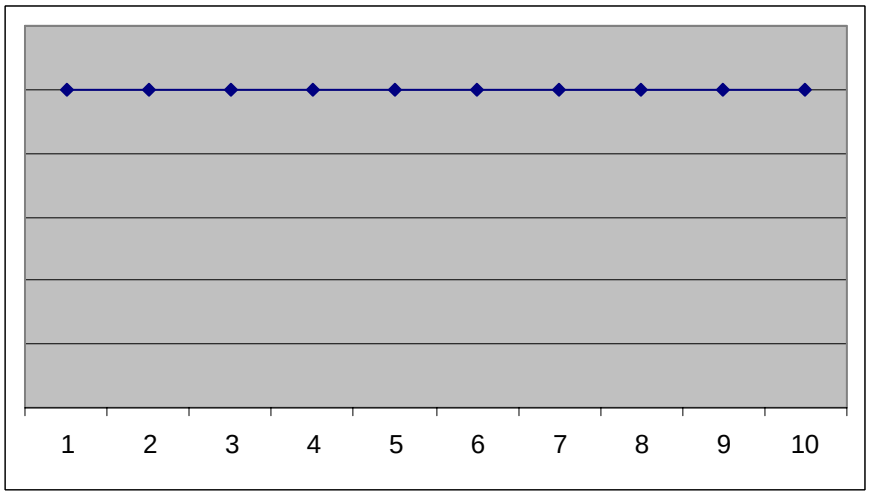
$$f(t) = \frac{dF(t)}{dt} = - \frac{dR(t)}{dt}$$

f(t)

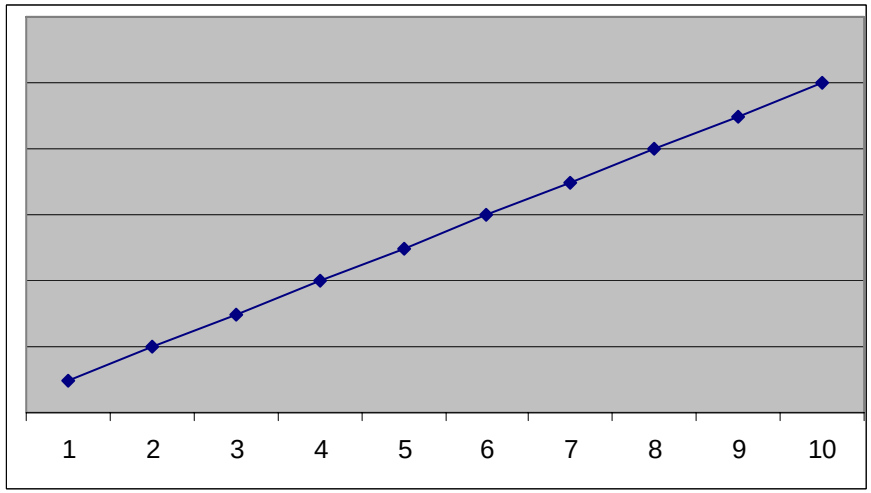


Konstante Ausfallwahrscheinlichkeitsdichte

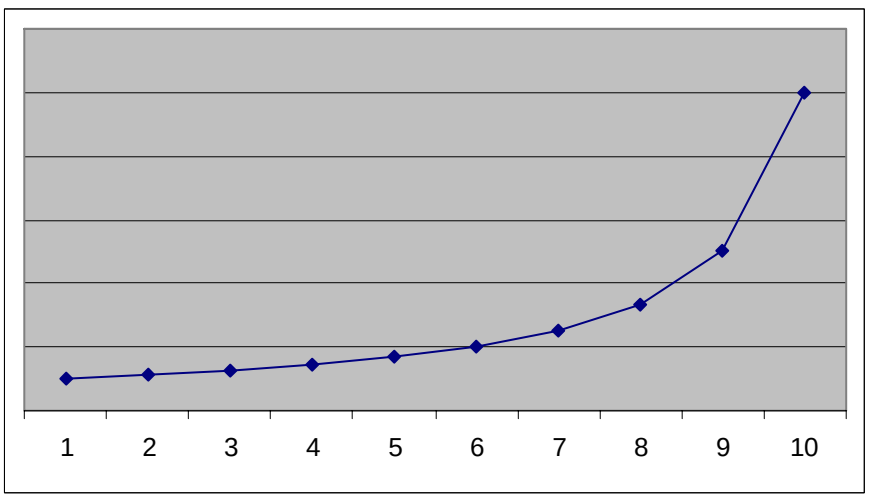
f(t)



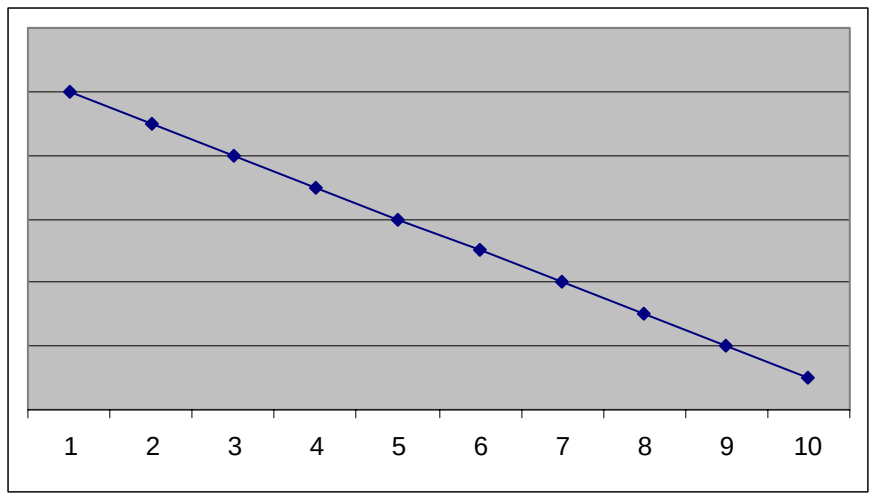
F(t)



Ausfallrate: λ

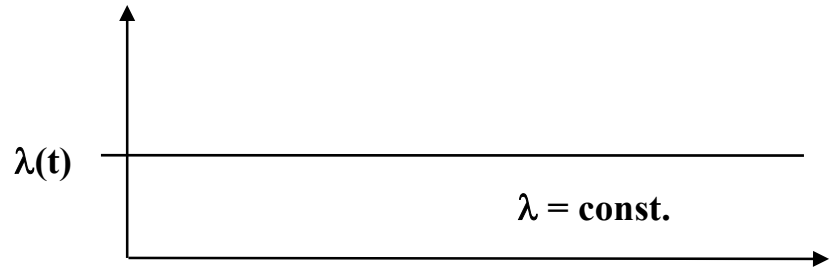


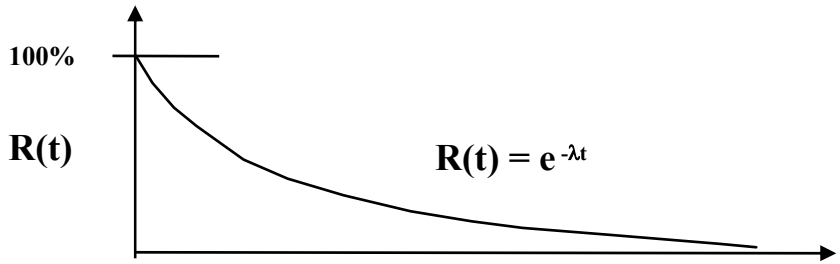
R(t)



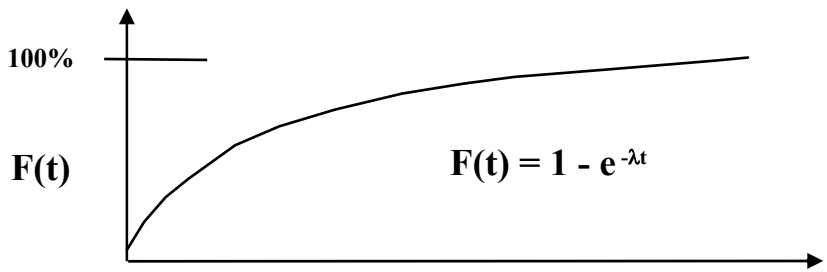
Maße der Fehlertoleranz

Ausfallrate λ (t)
Anzahl der Ausfälle pro Zeiteinheit

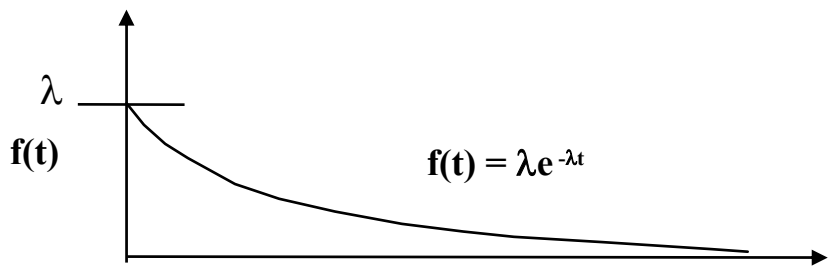




Bemerkung: **Die Ausfallrate ist relativ zum Bestand definiert. Fallen pro Zeiteinheit immer gleich viele Komponenten aus, steigt die Ausfallrate relativ zum Bestand an, der ja immer kleiner wird.**

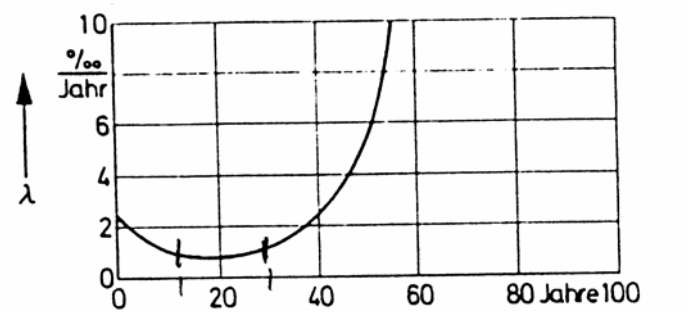


Bleibt die Ausfallrate relativ zum Bestand konstant, ergibt sich daraus eine Exponentialverteilung für die Überlebenswahrscheinlichkeit R(t).

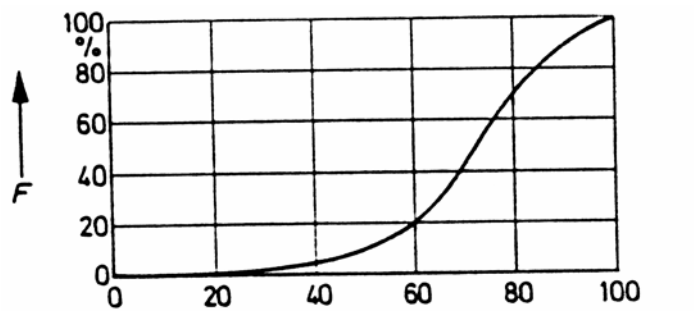


Lebensdauerverteilung beim Menschen

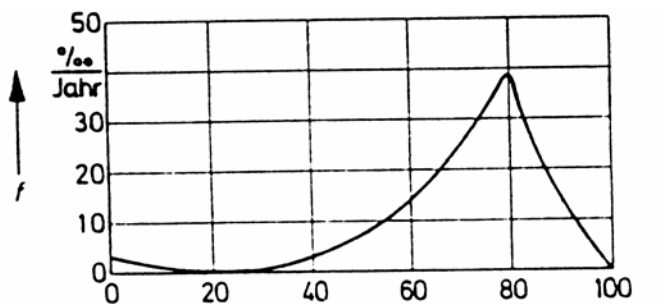
Ausfallrate λ (t)



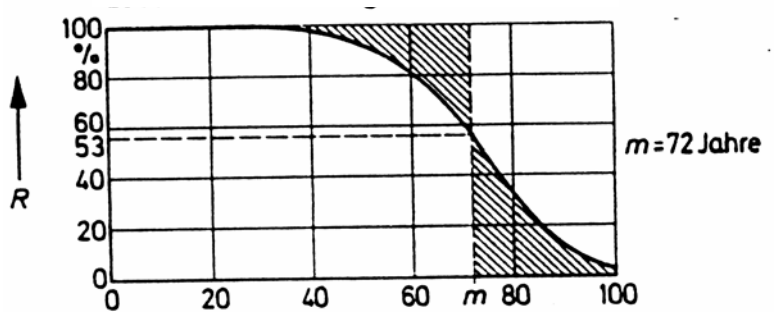
Ausfallwahrscheinlichkeit F(t)



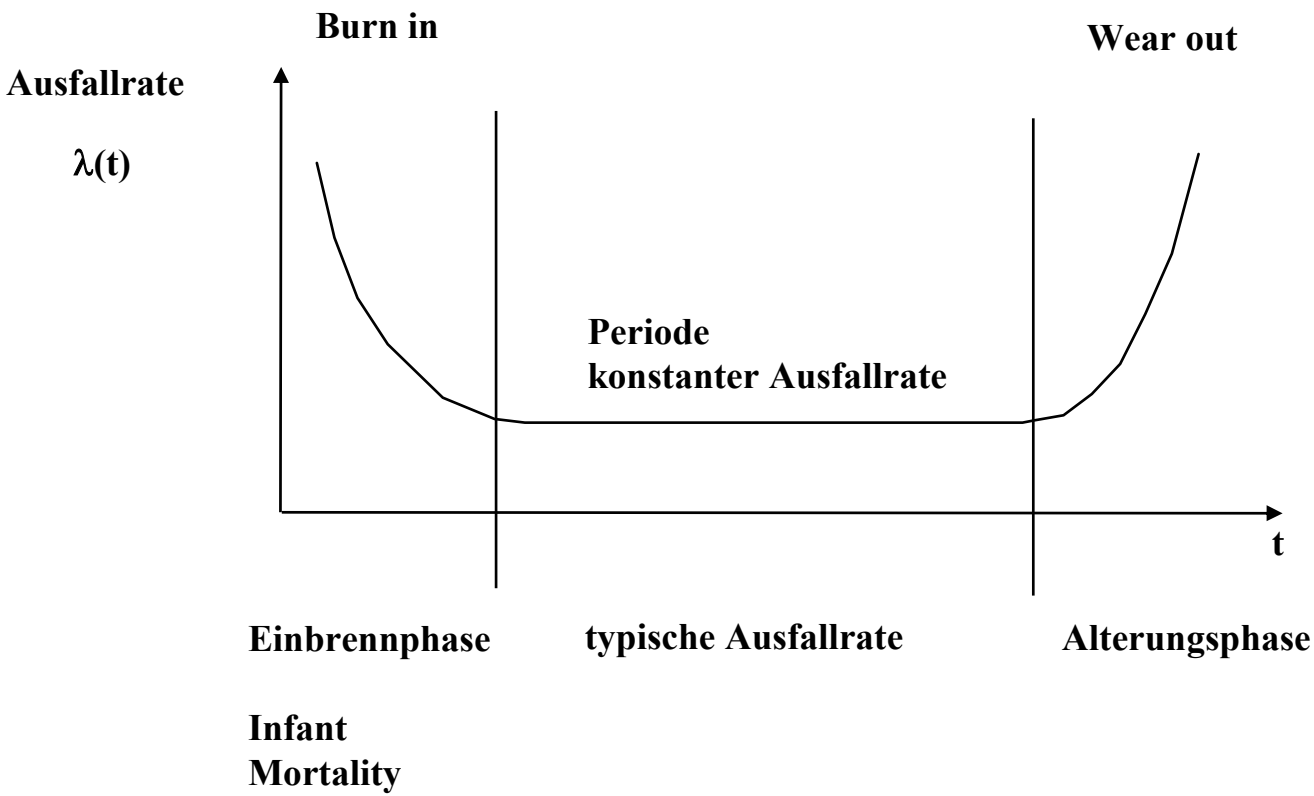
Ausfallwahrscheinlichkeitsdichte f(t)



Überlebenswahrscheinlichkeit R(t) (Reliability)



Maße der Fehlertoleranz (3)



Typische Ausfallrate:
VLSI-Chip: 10^{-8} Ausfälle/h = 1 Ausfall in 115000 Jahren

Kenngröße	Symbol	Einheit
Lebensdauer	T	h
Ausfallwahrscheinlichkeit	F	%
Überlebenswahrscheinlichkeit	R	%
Ausfallwahrscheinlichkeitsdichte	f	%/h
Ausfallrate	λ	$1/\mathrm{h}$

Maße der Fehlertoleranz

Unter der Annahme von $\lambda(t) = \text{const.}$ gilt:

$$\frac{1}{\lambda} = \text{MTBF} = \text{MTTFF} = \text{MTTF}$$

MTBF : Mean Time Between Failures

MTTFF: Mean Time To First Failure

MTTF : Mean Time To Failure

Maße der Fehlertoleranz

Verfügbarkeit (Availability)

Zeit in der ein System intakt ist bezogen auf die gesamte Missionszeit

$$A = \frac{\text{U (Up time)}}{\text{M (Mission time)}}$$

$$\text{M} = \text{U} + \text{TR (Repair time)}$$

$$A = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}$$

Maße der Fehlertoleranz (8)

Verfügbarkeit: Einteilung in Systemklassen

Klasse: $\lfloor \log_{10} (1/(1-A)) \rfloor$

1 Jahr = 525600 Minuten

Systemtyp	Nicht-Verfügbarkeit Minuten/Jahr	Verfügbarkeit %	Klasse
Nicht verwaltete Systeme	50 000	~ 90	1
Verwaltete Systeme	5 000	99	2
Gut verwaltete Syst.	500	99,9	3
Fehlertolerante Syst.	50	99,99	4
Hochverfügbare Syst.	5	99,999	5
Sehr hochverf. Syst.	0,5	99,9999	6
Ultra-hochverf. Syst.	0,05	99,99999	7

Maße der Fehlertoleranz

Beispiele:

- **Telefonvermittlungssysteme** **Klasse 5**
  Nichtverfügbarkeit 2h/a bis 3 Min /a

  Aber: in den letzten Jahren waren mehrere große Ausfälle:
  1 USA-nationsweiter Fehler: 8h
  1 Midwest: 4 Tage

- **Starkstromüberwachung** **Klasse 4**
  Nichtverfügbarkeit typ.: 2h/a

- **AAS (Advanced Automation System) IBM**
  3 sek/a für kritische Dienste (A = 0,9999999) **Klasse 7++**
  156 sek/a für weniger kritische Dienste (A = 0,9999950) **Klasse 5**

Anforderungen an die Autoelektronik

Anfangszuverlässigkeit:	**(0 *km* / 0 *h*) Fehler: < 500 • 10^{-9}**
	im 1. Jahr Fehler: < 1000 • 10^{-9}
System-Lebenszeit:	**3500 *h* (ca. 5Jahre bei 2*h*/Tag)**
Garantie:	**≥ 1 Jahr, Ersatzteile ≥ 10 Jahre**
Umgebungsbedingungen:	**-40 bis +85 ^oC**
Vibration:	**10 *Hz* bis 1 *kHz*, zufällig 5*g*, Sinus 2-5*g***
Shock:	**30 *g***
Versorgungsspannung:	**8-16*V***
	Motorstart mit 6*V* (-40 bis +85 ^oC), 18*V* für 2*h*, 24*V* für 1 *min*
	umgekehrte Polarität 13,5*V* für 1 *min*

Streßtest für Autoelektronik

- **Funktionstest: 8, 13.5, 16 *V* bei -40, 25, 85 ^oC**
- **Hitzetest: 85 ± 2 ^oC für 16*h* bei 16 *V* und 6000 *upm***
- **Kältetest: -40 ± 3 ^oC für 2*h***
- **Lagerung: 85 ± 2 ^oC für 504*h***
- **Temperaturschock: -40 bis 85 ^oC Übergang in 30 *sek.* 25 mal**
- **Temperaturänderung: -40 bis 85 ^oC , 3 ± 0.6 ^oC /*min* für 2 Zyklen**

Organisation der redundanten Komponenten

Arten der Redundanz:

- **Komponentenredundanz**
- **Zeitredundanz**

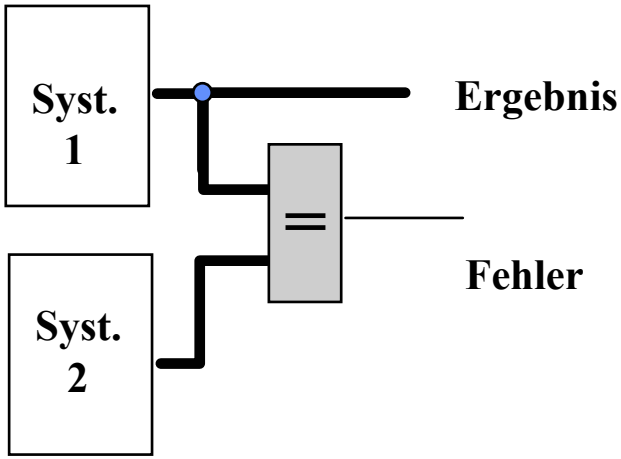
- **Aktive Redundanz: Mehrere Komponenten erbringen dieselbe Dienstleistung nebenläufig.**

- **Passive Redundanz: Redundante Komponenten sind nicht an der Erbringung der Dienstleistung beteiligt.**

- **Cold Standby: die redundante(n) Komponente(n) werden erst aktiviert, wenn eine aktive Komponente ausgefallen ist. Der Zustand der Berechnung zum Zeitpunkt des Ausfalls der aktiven Komponente muss auf der redundanten Komponente rekonstruiert werden.**

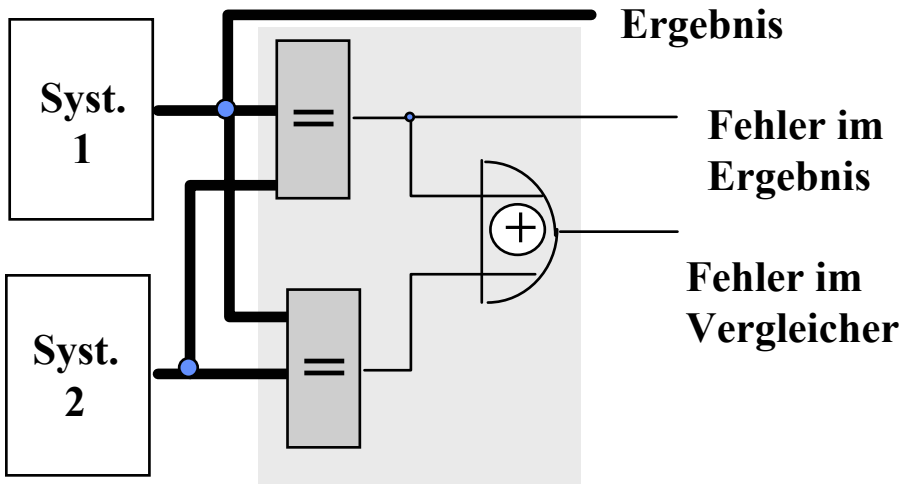
- **Hot Standby: die redundante(n) Komponente(n) ist aktiv, erzeugt aber keine Ausgaben. Die redundante Komponente enthält beim Ausfall der aktiven Komponente bereits deren Zustand und kann sie sofort ersetzen.**

Organisation der redundanten Komponenten

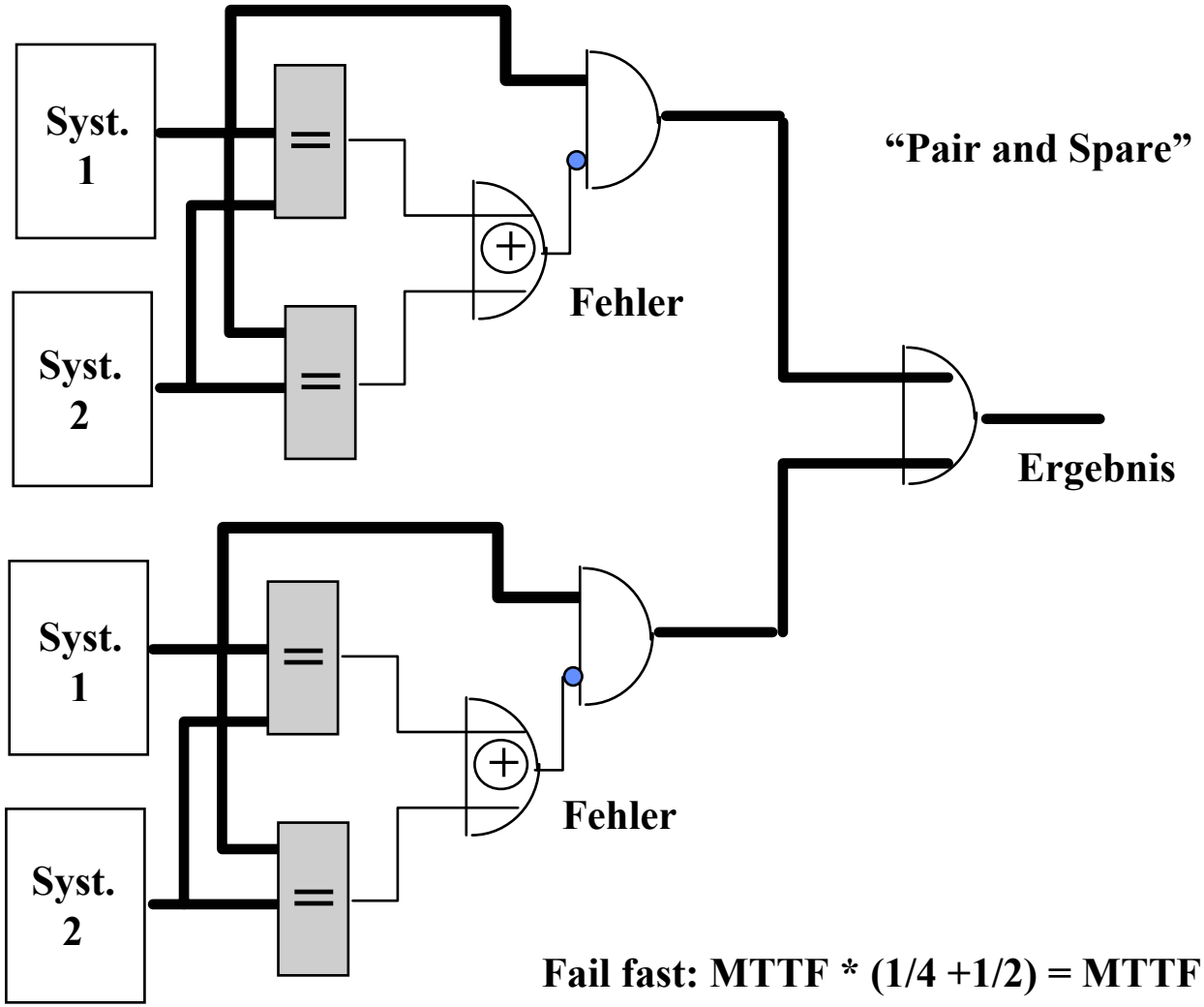


MTTF für ein Modul ist: 1* MTTF
Für ein System mit 2 Moduln gilt dann:
MTTF / 2.

Fail Fast: MTTF/2
Fail Soft: MTTF *(1+1/2) = MTTF * 3/2



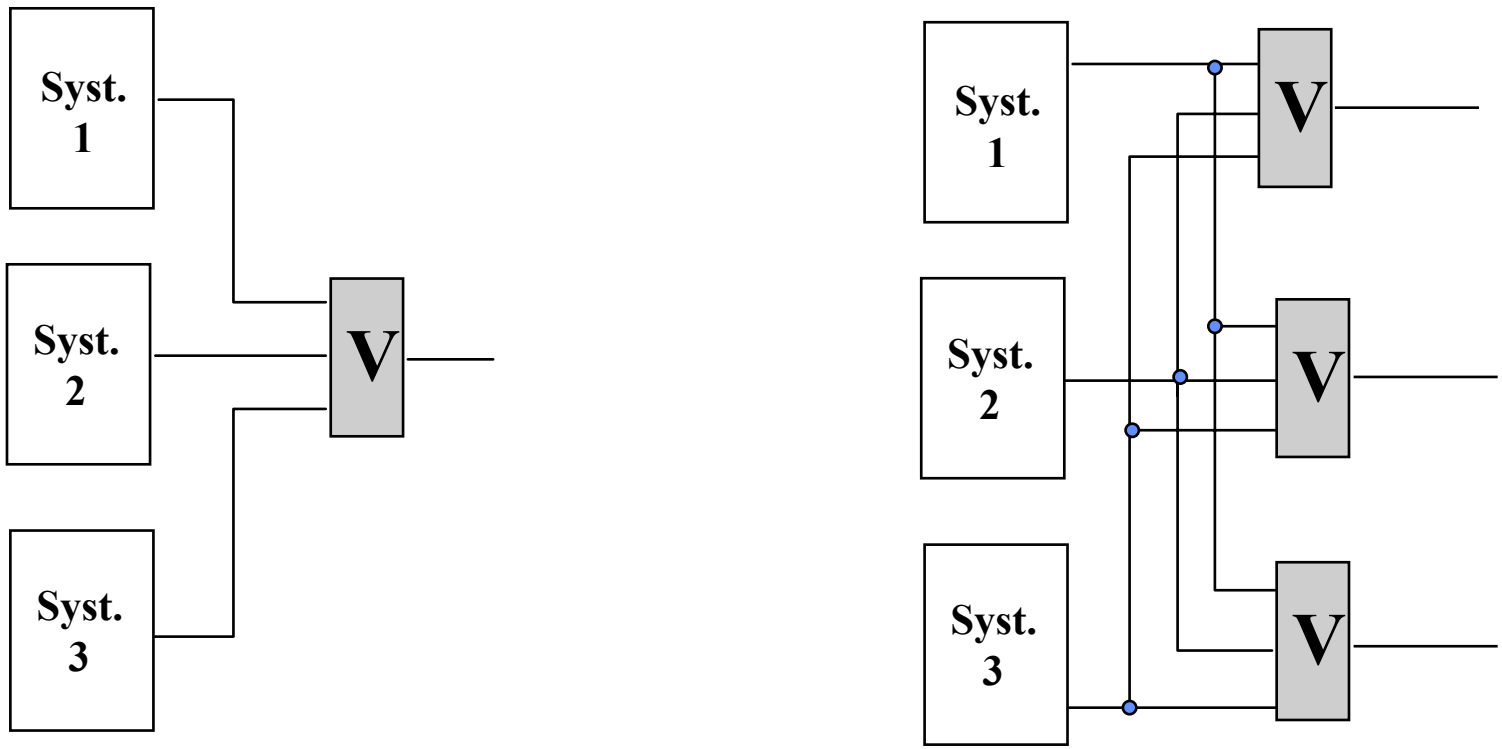
Organisation der redundanten Komponenten



Fail fast: MTTF * (1/4 +1/2) = MTTF * (3/4)
Fail soft: MTTF* ((3/4) + (2/4) + 1) = MTTF * 2,25

Organisation der redundanten Komponenten

Triple Modular Redundancy (TMR)



Fail Fast: MTTF/3 + MTTF/2 = MTTF (2/6) + MTTF (3/6) = MTTF(5/6)
Fail Soft: MTTF (1 + 5/6)

Organisation der redundanten Komponenten

Annahme: MTTF = 1 Jahr

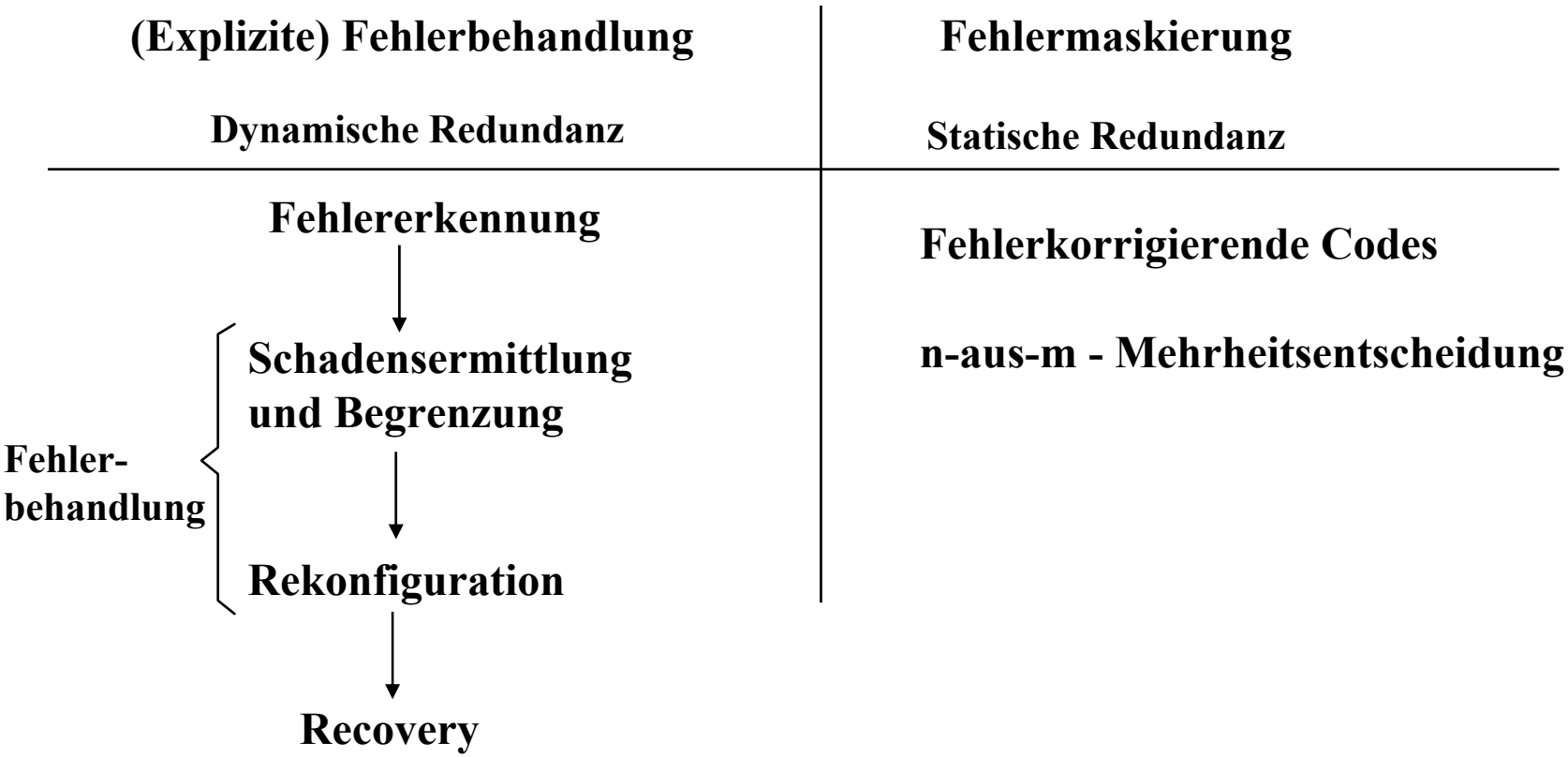
Organisation	MTTF (Jahre)	Klasse	Gleichung	Kosten
Simplex	1	3	MTTF * 1	1
Duplex (FF)	~ 0,5	3	MTTF/2	2 + ε
Duplex (FS)	~ 1,5	3	MTTF(3/2)	2 + ε
TMR (FF)	~ 0,8	3	MTTF(5/6)	3 + ε
TMR (FS)	~ 1,8	3	MTTF(1+5/6)	3 + ε
Pair&Spare (FF)	~ 0,7	3	MTTF(3/4)	4 + ε
TMR + Repair	$> 10^6$	6	$MTTF^3 / 3\ MTTR^2$	3 + ε
P&S, FS + Repair	$> 10^3$	6	$MTTF^2 / 2\ MTTR$	4 + ε

FF: Fail fast
FS: Fail soft

Quelle: J.Gray

Mechanismen der Fehlertoleranz

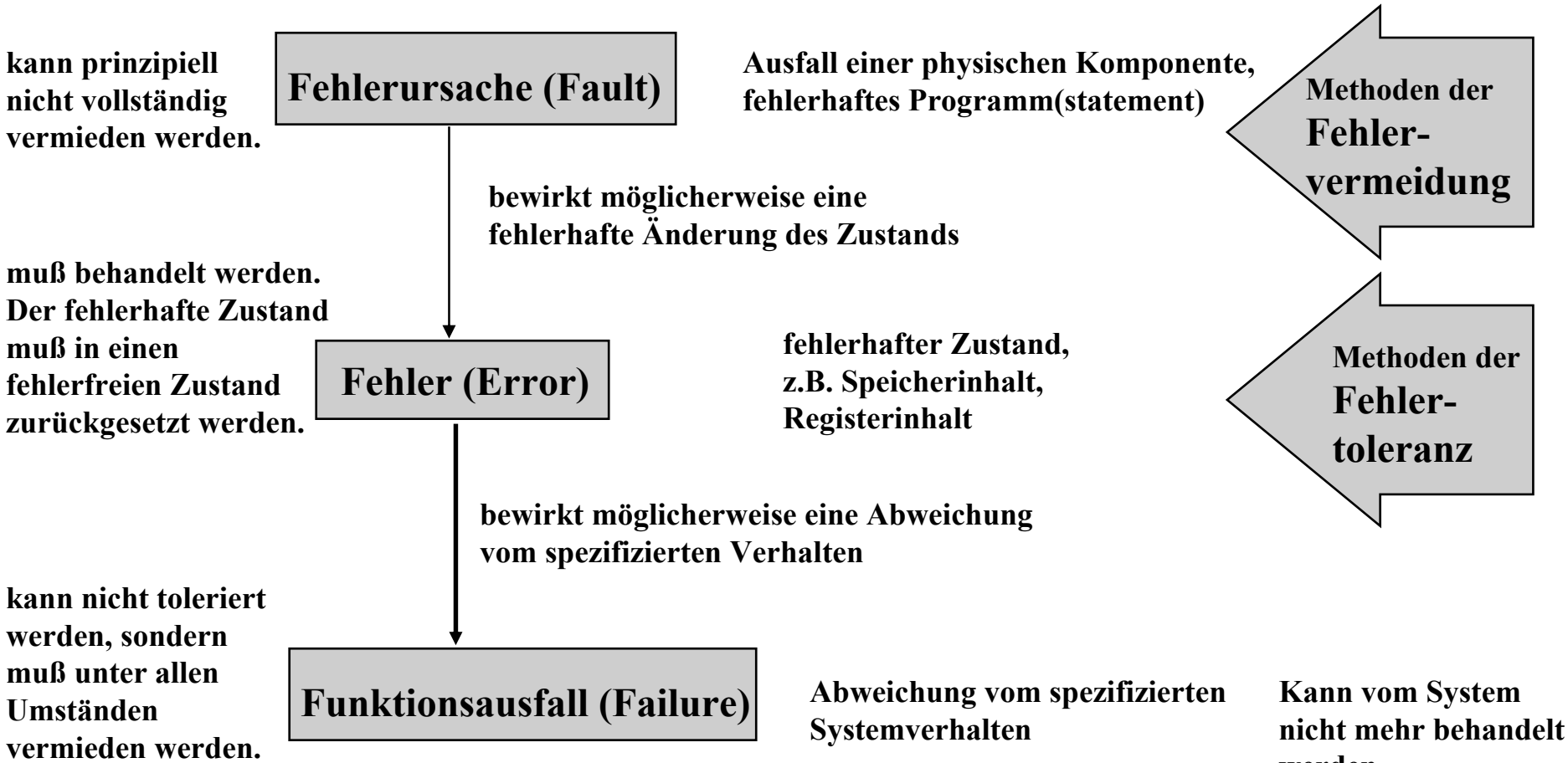
Mechanismen der Fehlertoleranz



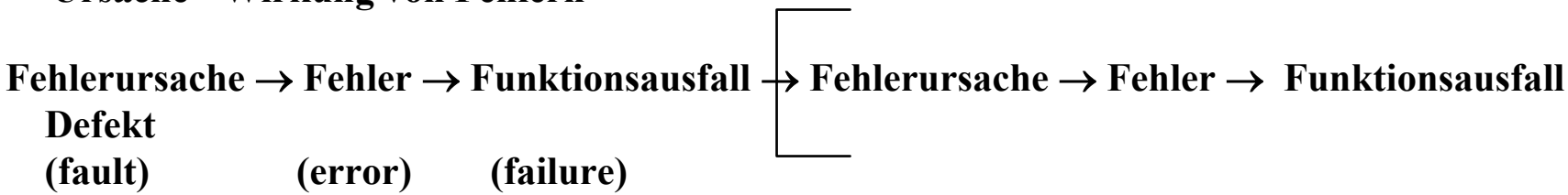
Alle Mechanismen der Fehlertoleranz beruhen auf Redundanz

- **Informationsredundanz**
- **Komponentenredundanz**
- **Zeitredundanz**

Fehlerklassifizierung



Ursache - Wirkung von Fehlern



Defekt: Ereignis
Fehler: Auswirkungen der Fehlerursache auf den Systemzustand
Funktionsausfall: Abweichung des Systems von seinem spezifizierten Verhalten

Defekt → Fehler

- ein Defekt, der durch den Berechnungsvorgang (noch) nicht aktiviert wurde, heißt *ruhend (dormant)*.
- Ein Defekt ist *aktiv*, wenn er einen Fehler verursacht.

Fehler → Funktionsausfall

- ein Fehler heißt *latent*, wenn er noch nicht zu einem Funktionsausfall geführt hat (oder noch nicht durch Erkennungsmaßnahmen entdeckt wurde).
- ein Fehler heißt *effektiv* , wenn er zu einem Funktionsausfall führt.

Funktionsausfall → Defekt

- ein Funktionsausfall tritt ein, wenn ein Fehler effektiv wird und den erbrachten Service verfälscht.
- ein Funktionsausfall kann die Fehlerursache für eine höhere Systemebene darstellen.

Mechanismen der Fehlererkennung

Code-Checks

basieren auf einem fehlererkennenden Code.

Diagnostische-Checks

überprüfen einzelne Komponenten des Systems, indem sie aus der Struktur der Komponente Eingabewerte ableiten, die antizipierbare Fehler in der Komponente aktivieren und zum Ausgang propagieren. Die Komponenten werden dann anhand des bekannten Ein/Ausgabeverhaltens getestet.

Timing-Checks

überprüfen, ob eine bestimmte Komponente festgelegte Zeitbedingungen einhält.

Reversive-Checks

berechnen aus den Ergebnissen die möglichen Eingabewerte und vergleichen sie mit den tatsächlichen Werten.#

Replikations-Checks

basieren auf einer Vervielfältigung einer Aktivität und anschließendem Vergleich der Ergebnisse.

Plausibilitäts-Checks

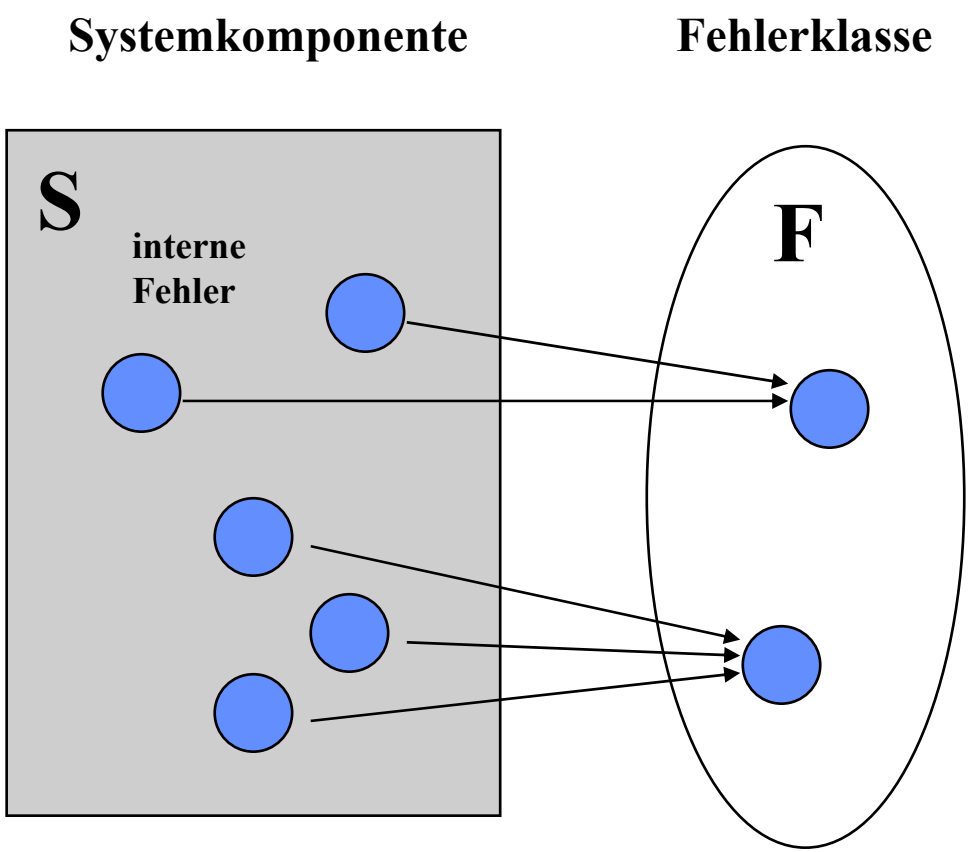
überprüfen den Zustand von (abstrakten) Objekten oder die Ergebnisse einer Berechnung auf ihre Plausibilität hinsichtlich der beabsichtigten Nutzung und des Zwecks dieser Objekte.

Strukturelle-Checks

überprüfen die strukturelle Integrität von Datenstrukturen.

Fehlersemantik

Die Fehlersemantik beschreibt die Annahmen über die Auswirkungen interner Fehler auf das beobachtbare Verhalten einer Systemkomponente (Abstraktion interner Fehler).



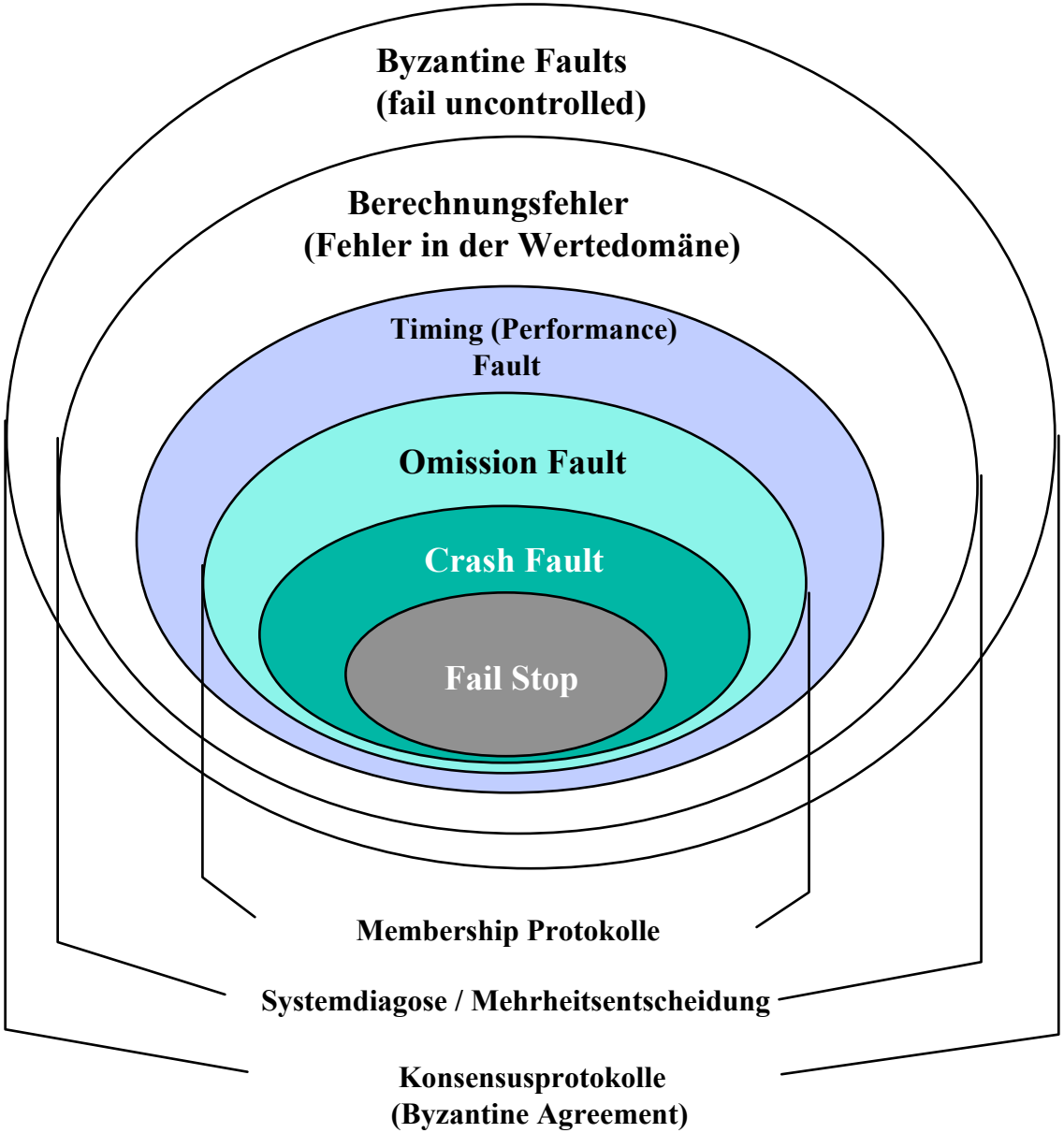
S hat die Fehlersemantik F

Problem:
Die Mechanismen zur Fehlerbehandlung beziehen sich auf die antizipierte Fehlerklasse.

Es muss gewährleistet werden, dass die Fehlersemantik F im System auch durchgesetzt wird.

Beispiele:
Omission-Fehlersemantik
Crash-Fehlersemantik

Hierarchie der Fehlermodi



Byzantinische Fehler:
Beliebige, unkontrollierbare Fehler

Zeitfehler.
Korrekte Resultate in der Wertedomäne, aber zu früh oder zu spät.

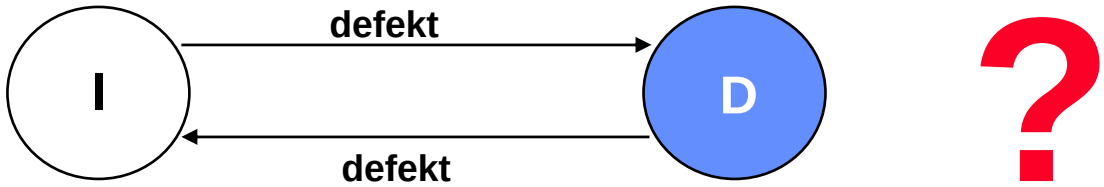
Omission (Unterlassungs-) Fehler:
Spezielle Klasse der Zeitfehler. Die (korrekten) Resultate kommen entweder zur richtigen Zeit oder gar nicht.

Crash Fehler:
Komponente liefert keine Resultate mehr.

Fail Stop:
Andere Komponenten können den Crash Fehler korrekt diagnostizieren.

Wie Testen ?

Systemdiagnose zur Fehlererkennung



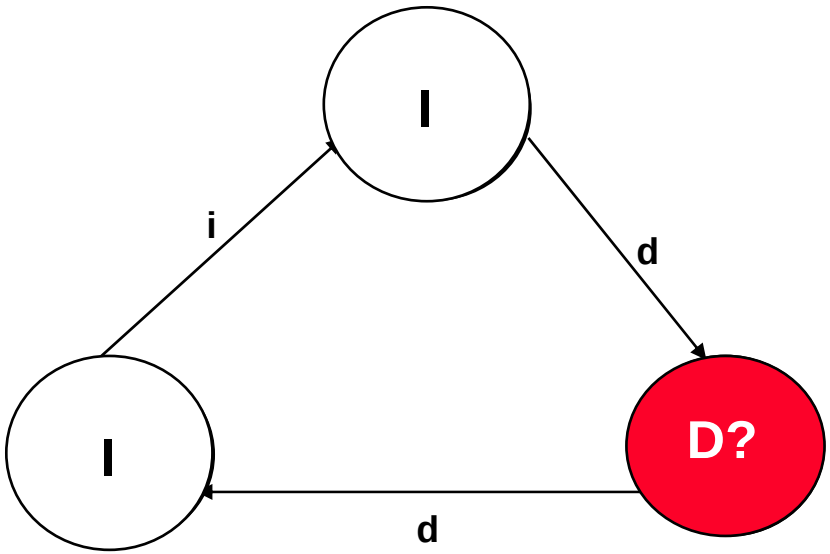
Annahmen:

- **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



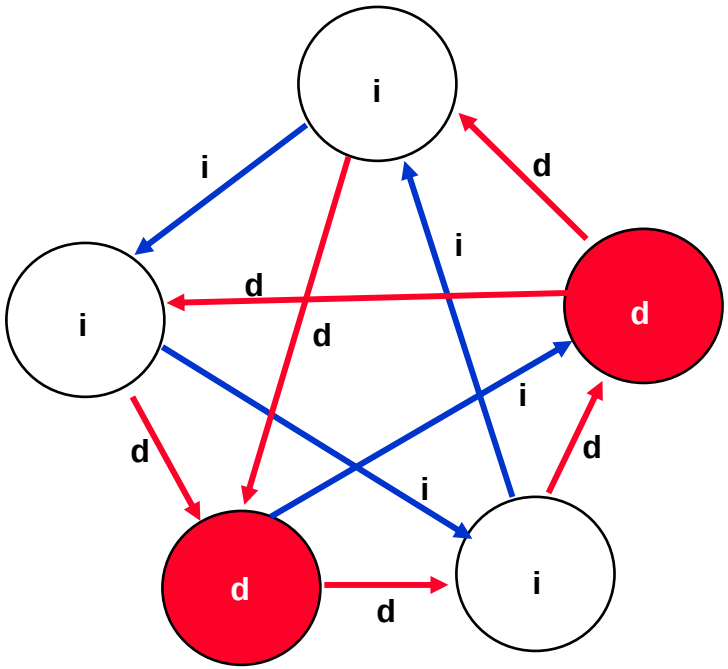
Annahmen:

- **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:

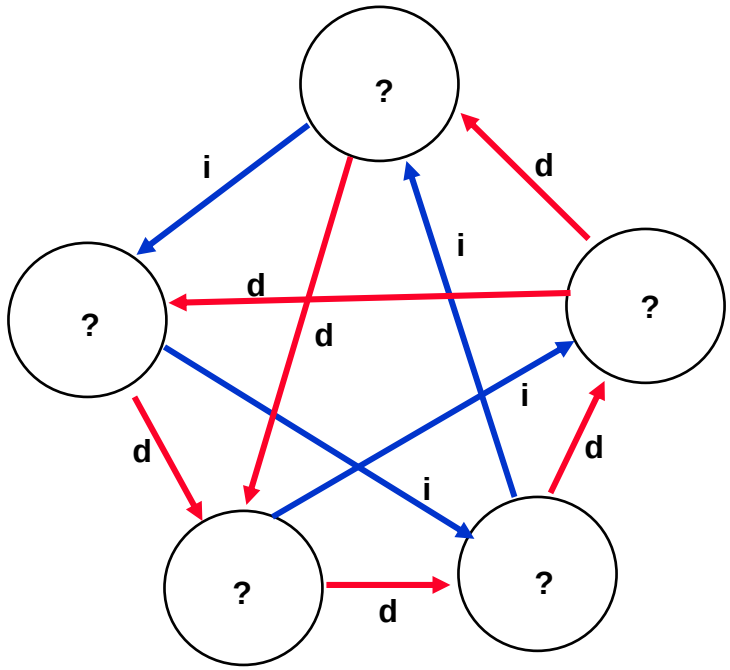
Ein System mit n Komponenten ist f–diagnostizierbar wenn $n \geq 2f + 1$ und jede Komponente mindestens f Komponenten testet, wobei sich die Komponenten nicht gegenseitig testen.

2-diagnostizierbares System



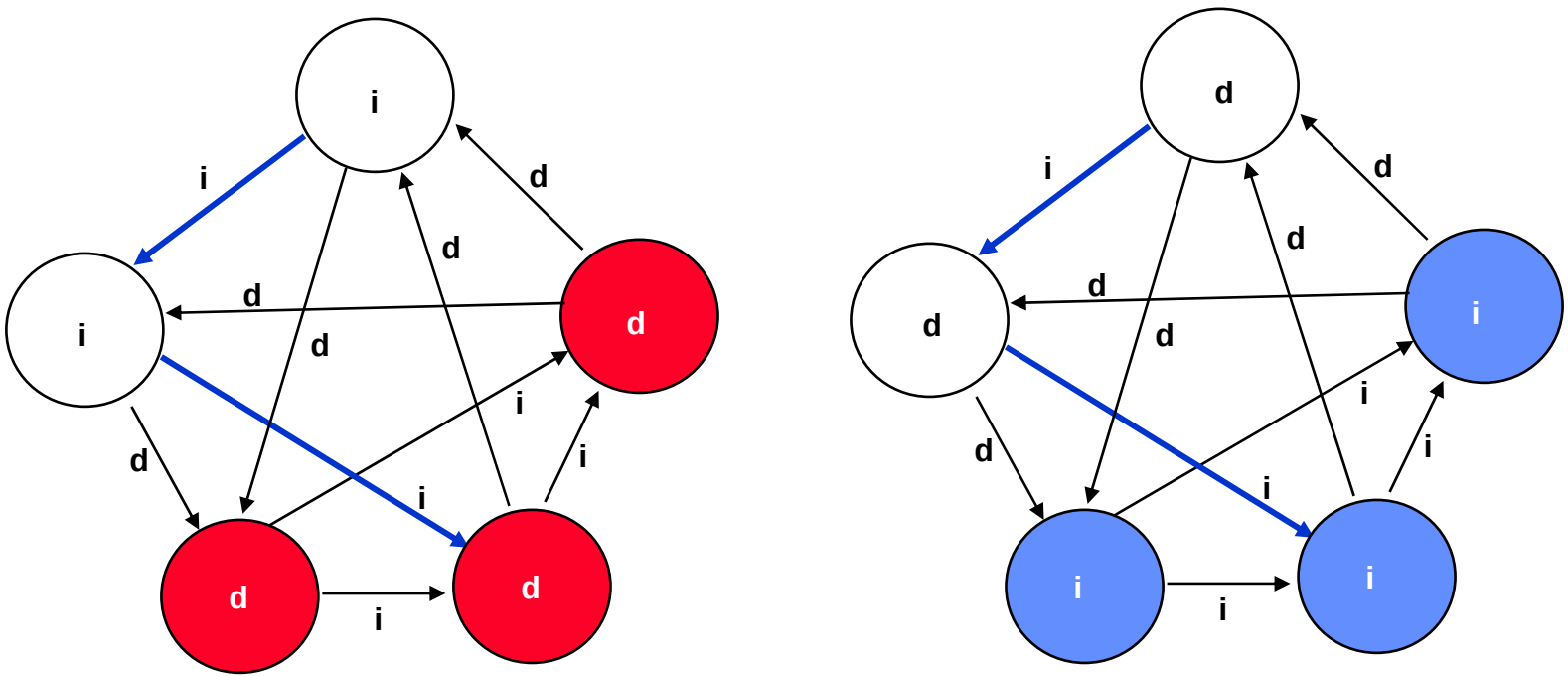
Annahmen:

- **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):

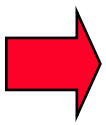
Strenge Konsistenz (SK): Ein korrekter Prozeß wird nie als ausgefallen erkannt. (Sicherheitskriterium)

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

Unter welchen Bedingungen können SK und SV erreicht werden ?

Annahmen:

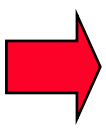
1. **Die Laufzeit der Nachrichten ist beschränkt,**
2. **Die Prozesse können ein Lebenszeichen in einem beschränkten Zeitintervall erzeugen.**
3. **Fehlermodell 1**



„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.**

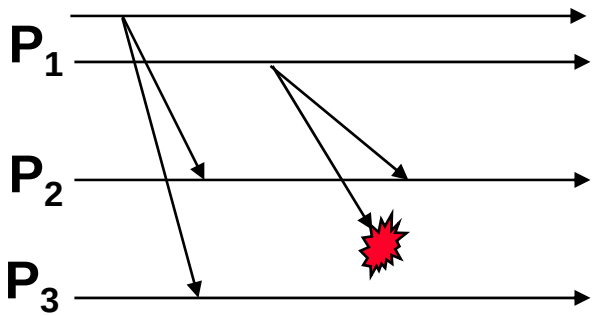


Einsatz von Mechanismen zur Maskierung von Omissions

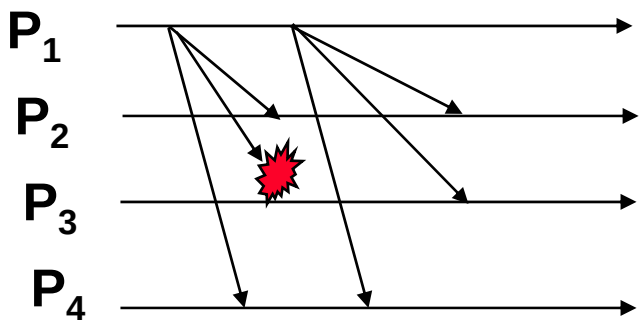
FT Kommunikation - Behandlung von Nachrichtenfehlern:

Statische Redundanz: Maskierend

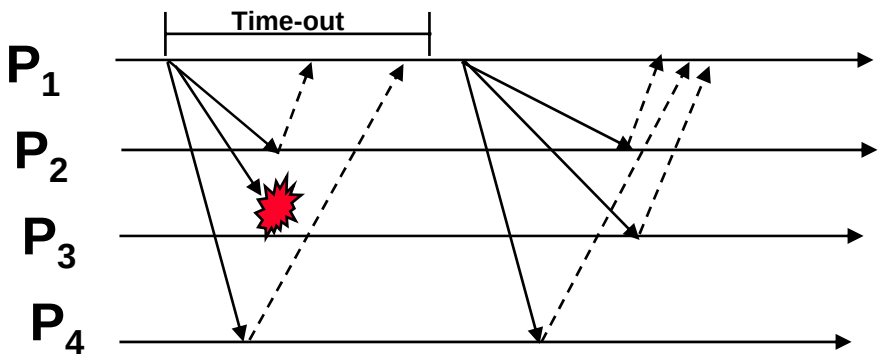
Komponentenredundanz



Zeitredundanz

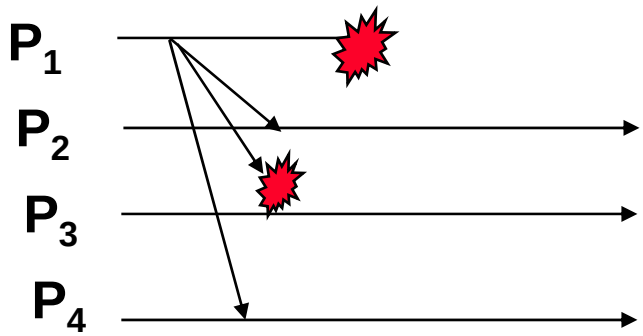


Dynamische Redundanz: Erkennung, Recovery

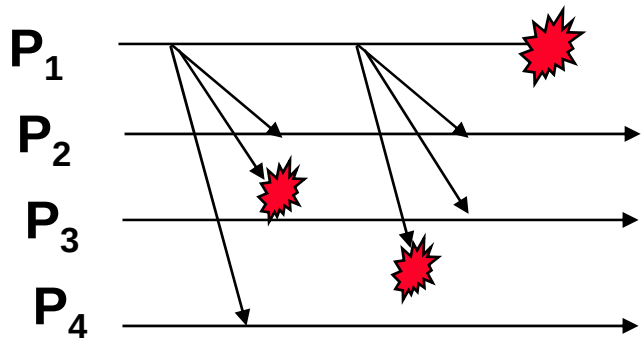


FT Kommunikation - Behandlung von Senderfehlern:

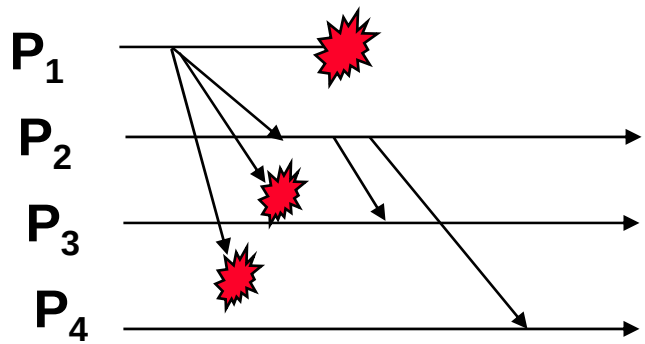
Unzuverlässiger Multicast



Best effort Multicast



Zuverlässiger Multicast



Nicht perfekte Fehlererkennung

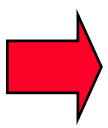
Annahmen:

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.**

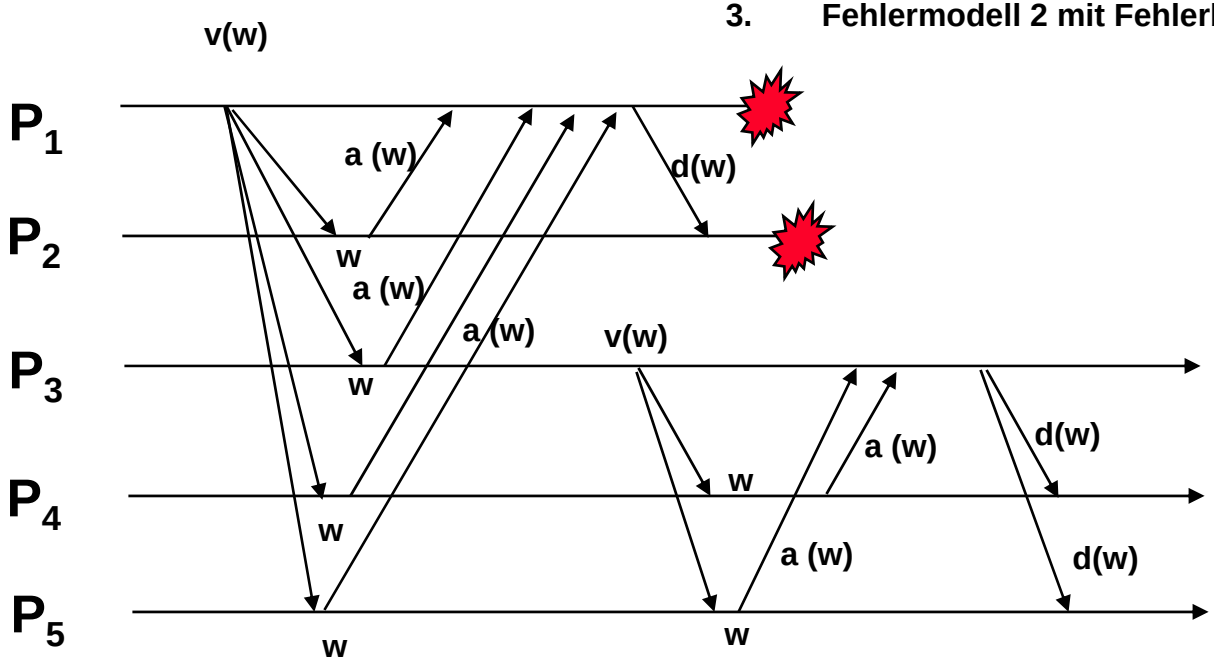
Nicht-Trivialität: **Der Wert, auf den sich geeinigt wurde muß der Eingabewert eines Prozesses sein (oder eine Funktion dieses Eingabewertes).**

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**

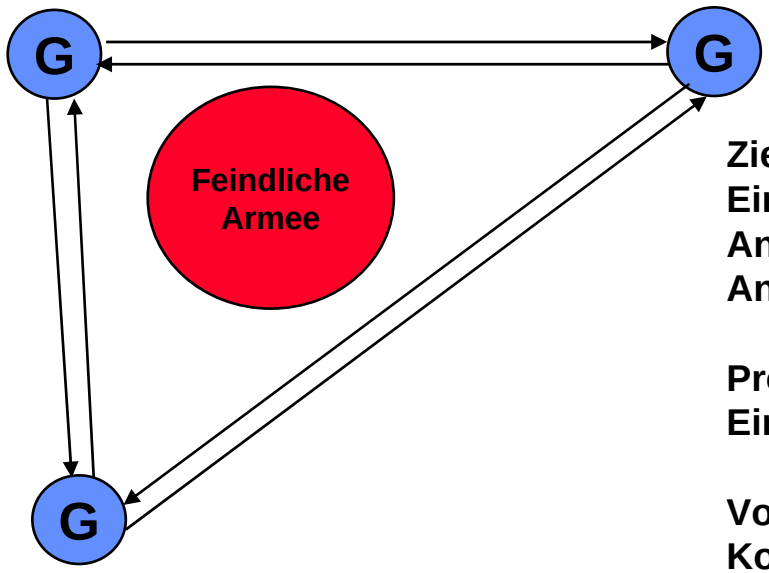


v(w): vorschlagen(w)
a(w): akzeptiert (w)
d(w): entschieden (w)

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:

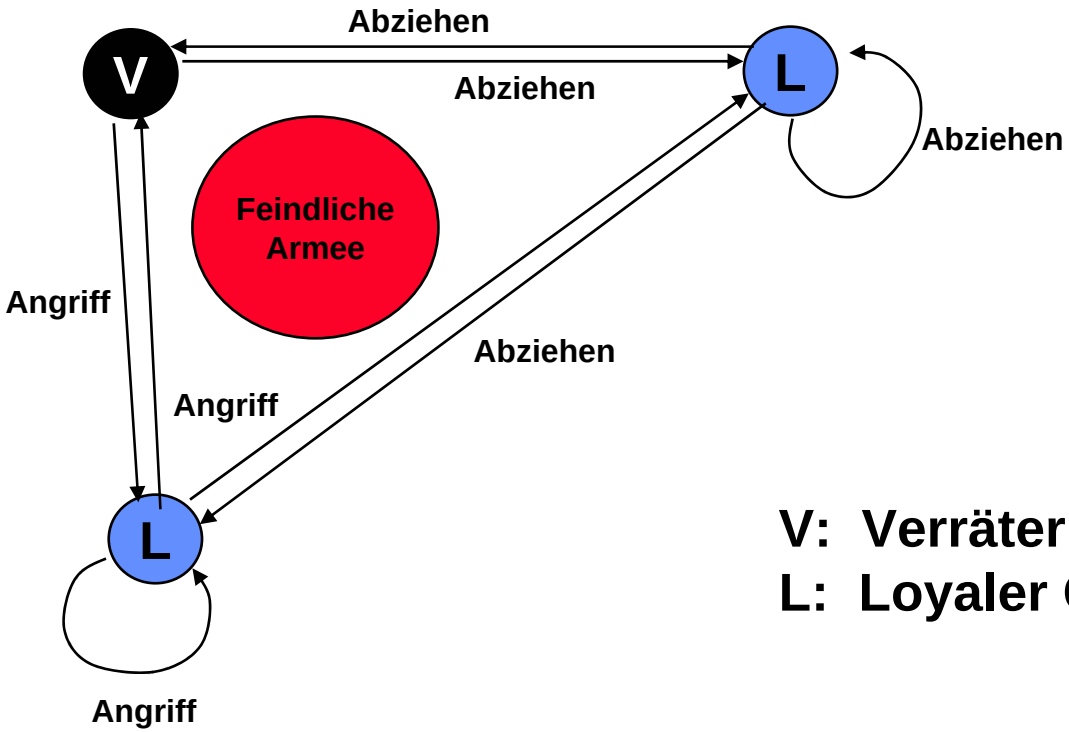


Ziel:
Einigung auf einen gemeinsamen Plan, Angreifen oder Abziehen? Nur bei einem gemeinsamen Angriff kann man erfolgreich sein.

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?

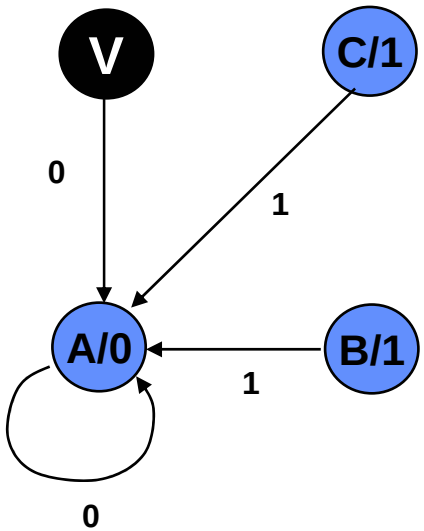


V: Verräter
L: Loyaler General

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

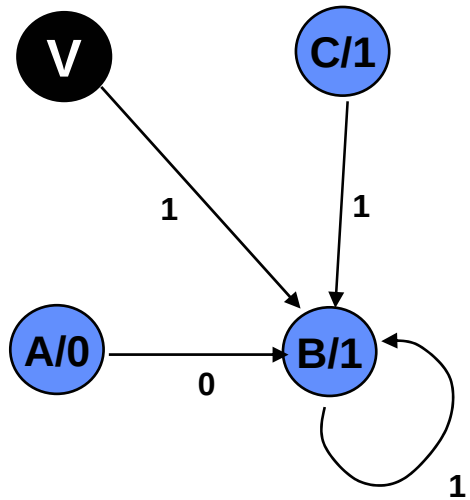
Einigung auf einen Wert in zwei Runden:

Nachrichten, die A erreichen



1. Runde

Nachrichten, die B erreichen

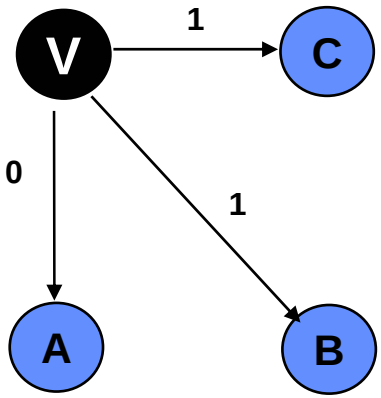


Verteilung der Werte

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

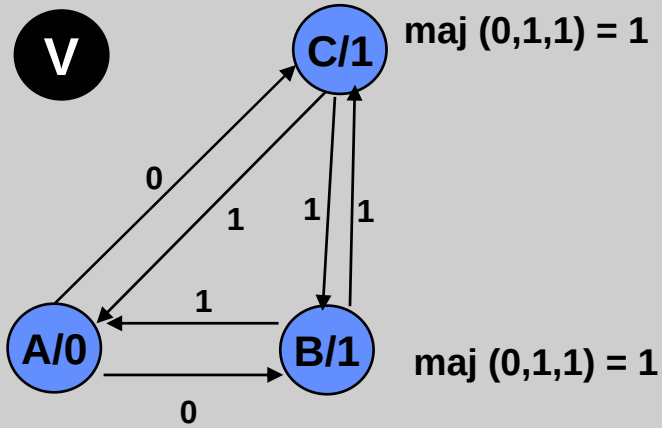
1. Runde

Verteilen der Werte von einem Teilnehmer



2. Runde

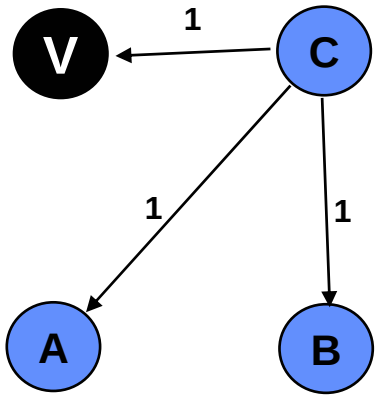
Einigung auf einen Wert, den ein Teilnehmer geschickt hat.



maj (0,1,1) = 1

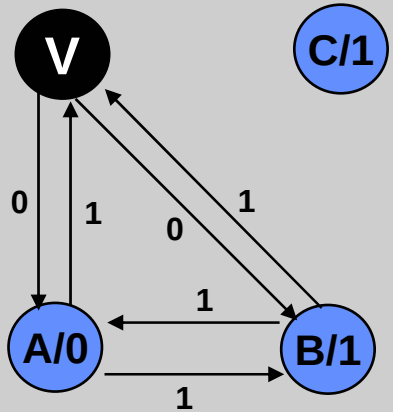
1. Fall

Sender ist Verräter



2. Fall

Verräter verfälscht bei Weiterleitung



maj (0,1,1) = 1

maj (0,1,1) = 1

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.