Sichere und fehlertolerante verteilte Systeme: Wunsch oder Wirklichkeit?

Felix Gärtner

fcg@acm.org

Korrekte Systeme?

Quelle: http://www.mpia-hd.mpg.de/SUW/SuW/1996/10-96/S713Abb1.html

Es funktioniert nicht, obwohl alles so ist wie erwartet!

Fehlertolerante Systeme?

Quelle: http://www.1a-homepage-werbung.de/wahnwitz/Atomkraft-Hysterie/chern2.jpg

Es funktioniert nicht, trotz redundanter Steuerungen!

Sichere Systeme?

Quelle: http://www.rp-online.de/news/multimedia/netzreporter/2001-0115/

Es funktioniert nicht, trotz Virenscanner!

Kritische Infrastrukturen

Quelle: http://www.cs.virginia.edu/~survive/

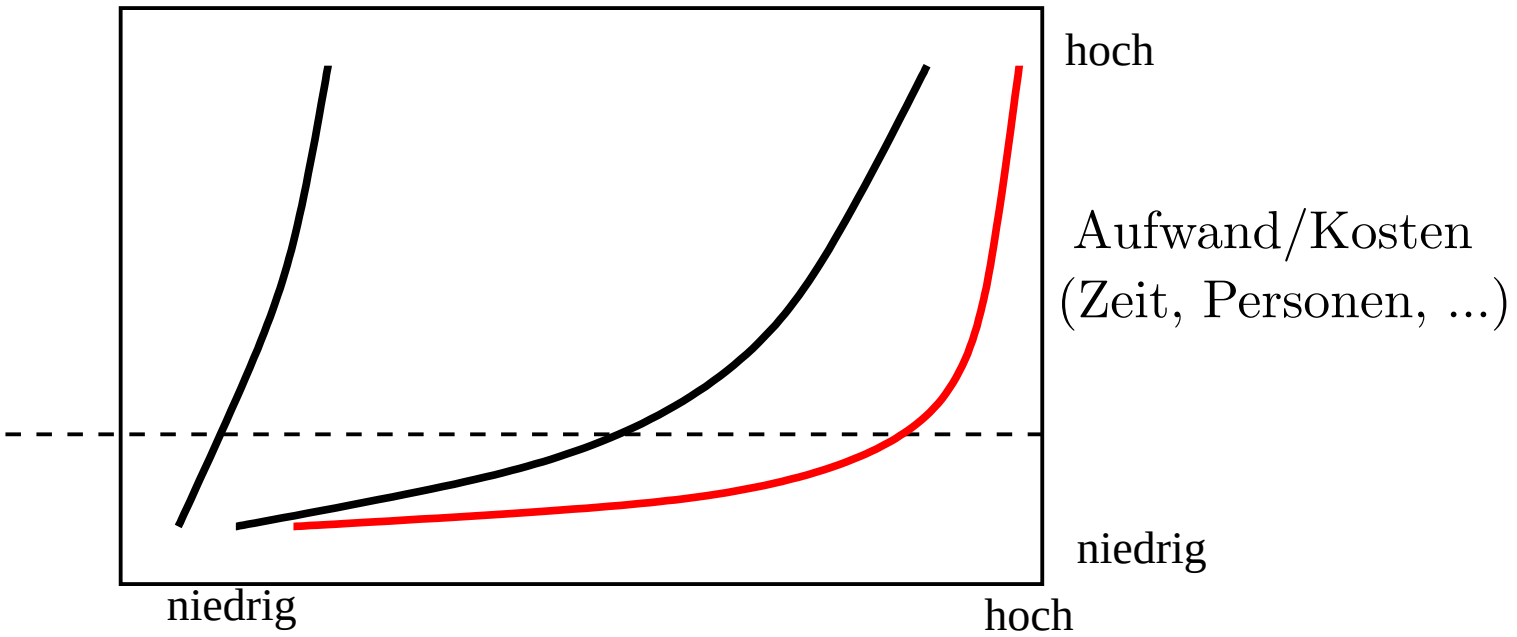
- Computersysteme schaffen Flexibilität.
- Gefahren für die Gesellschaft unabsehbar.
- U.S.-Strategen beschwören bereits “elektronisches Pearl Harbor.”

Verläßlichkeit

- Eigenschaften verläßlicher Systeme (*dependable systems* [Laprie 1992]):
  - Verfügbarkeit (*availability*)
  - Zuverlässigkeit (*reliability*)
  - Sicherheit (*safety* und *security*)
- Verläßliche Systeme zu bauen ist schwer! [Neumann 1995]
- Hohe Komplexität, hoher Aufwand, hohe Kosten.

Wissenschaftliche vs. industrielle Interessen

- Erfahrung: Mehr Rigorosität $\approx$ mehr Verläßlichkeit.
- Erfahrung: Mehr Rigorosität $\approx$ höhere Kosten.



Ziel: Theoretisch abgesicherte Ingenieursmethodik

- Bestandteile:
  - Präzise und genau verstandene Systemmodelle.
  - Realistische Fehler-/Angreifermodelle.
  - Korrekte Entwurfstheorien/-muster.
  - Nützliche vorgefertigte Grundbausteine.
- "Angebote an die Ingenieursintuition."

Gliederung

- Motivation
- Thema: Fehlertoleranz
  - Beiträge der Wissenschaft
  - Einsichten für Ingenieure
- Thema: *security*
  - Beiträge der Wissenschaft
  - Einsichten für Ingenieure
- Zusammenfassung und Ausblick

Fehlertoleranz

Beiträge der Wissenschaft

Asynchrones Systemmodell

- Verteiltes System aus Rechnerknoten, die mittels Nachrichtenaustausch kommunizieren.
- Nachrichten können beliebig lange unterwegs sein.
- Rechnerknoten können beliebig langsam sein.
- Gebräuchliches Systemmodell für das Internet.
- Effekte verschiedener Systemparameter (Fehler, Randomisierung, Uniformität) recht gut verstanden.

Eigenschaften verteilter Systeme

- System Σ: nichtdeterministische Zustandsmaschine
- Eigenschaft P: Menge von Abläufen $\sigma = s_1, s_2, \ldots$
  ($sem(\Sigma) =$ alle Abläufe des Automaten)
- Sicherheitseigenschaft S ("immer . . . "):

$$\sigma \notin S \Rightarrow \exists i. \forall \beta. \sigma|_i \cdot \beta \notin S$$

- Lebendigkeitseigenschaft L ("schlußendlich . . . "):

$$\forall i. \exists \beta. \sigma|_i \cdot \beta \in L$$

Fehlertoleranzspezifikationen [Arora and Kulkarni 1998; Gärtner 1999]

- Eigenschaftsklassen sind fundamental [Alpern and Schneider 1985]: $\forall P.\exists S, L.P = S \cap L$
- Fehler können zu einer Verletzung von S oder L führen.
- Arten von Fehlertoleranz:
  - Maskierend: $P' = S \cap L$
  - Nicht-Maskierend: $P' = \Diamond S \cap L$
  - Fail-Safe: $P' = S$

Fehlertoleranzkomponenten

- Fehlertolerantes Programm = fehler-intolerantes Programm + Fehlertoleranzkomponenten
- Detektor: erkennt Systemzustand
- Korrector: erzwingt Systemzustand
- Abstraktionen von vielen bekannten Fehlertoleranzmechanismen, gute Strukturierungsmöglichkeit [Gärtner 1998].

Fehlertoleranztheorie

- Theoreme [Arora and Kulkarni 1998]:
  - Detektoren sind notwendig und hinreichend für Sicherheitseigenschaften.
  - Korrektoren sind notwendig und hinreichend für Lebendigkeitseigenschaften
- Funktionsprinzip: Redundanz [Gärtner and Völzer 2001]
  - Platzredundanz = nicht-erreichbarer Zustand
  - Zeitredundanz = nicht-ausführbare Transition

Notwendigkeit von Platzredundanz

- $\Sigma = (C, I, T, A)$, Fehlermodell F, Eigenschaft P.
- Σ erfüllt P, $F(\Sigma)$ verletzt P.
- Baue Σ' aus Σ so daß $F(\Sigma')$ erfüllt P.
- Dann hat Σ' nicht erreichbare Zustände.
- Beweis: Σ' besitzt eine Transition, die auch in Σ zur Verletzung von P geführt hätte.

Fehlertoleranz

Beiträge zur Ingenieursmethodik

Allgemeine Fragestellungen

- Welches Systemmodell sollte verwendet werden?
- Welche Art von Fehlertoleranz soll erreicht werden (maskierend, nichtmaskierend, fail-safe)?
- Welches Fehlermodell wird in welcher Komponente angenommen?
- Welche Auswirkungen auf die Systemeigenschaften haben diese Fehlermodelle (Lösbarkeit, Menge der Redundanz)?

Konkrete Fragestellungen

- Wo benötige ich Detektoren, wo Korrektoren?
- Wie baue ich Detektoren und Korrektoren schrittweise in das System ein?
- Wie arbeite ich redundanzoptimal?
- Wie hoch ist die Wahrscheinlichkeit eines Ausfalls?

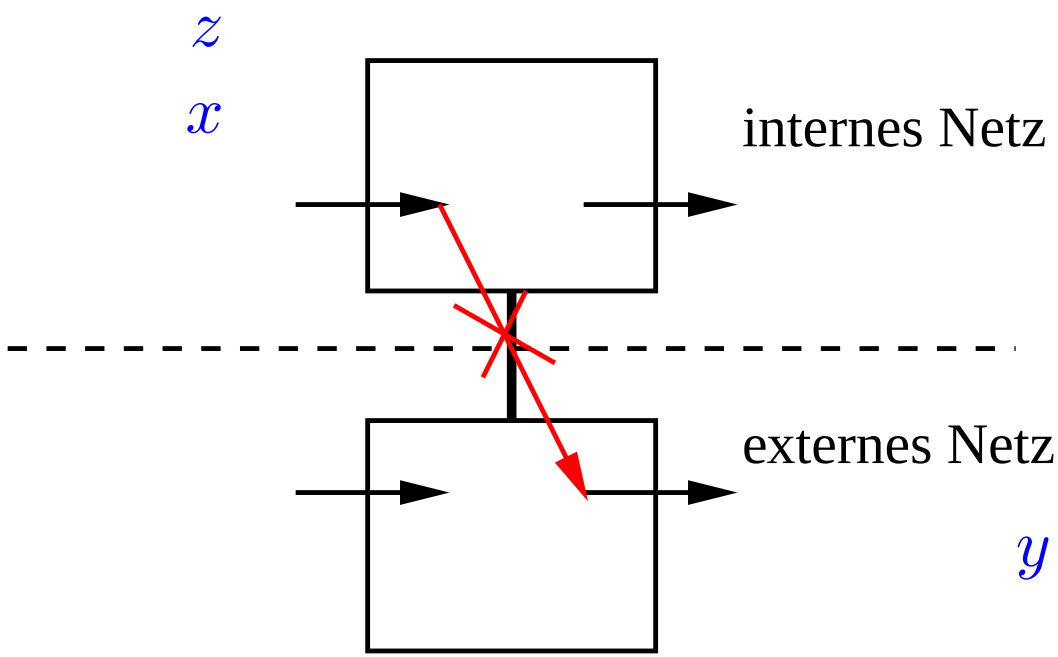
Security

Wissenschaftliche Perspektive

Was ist *security*?

- Auch Sicherheit- und Lebendigkeit:
  - Sicherheitseigenschaft: Zugriffskontrolle [Schneider 1998], Vertraulichkeit [Gray, III. and McLean 1995], Integrität.
  - Lebendigkeitseigenschaft: Verfügbarkeit.
- Probleme mit Informationsfluß über verdeckte Kanäle (relevant in Hochsicherheitsbereichen), z.B. träge Firewall.

Informationsflußeigenschaften



Mögliche Abläufe: x, y z, y

Eigenschaften von Eigenschaften

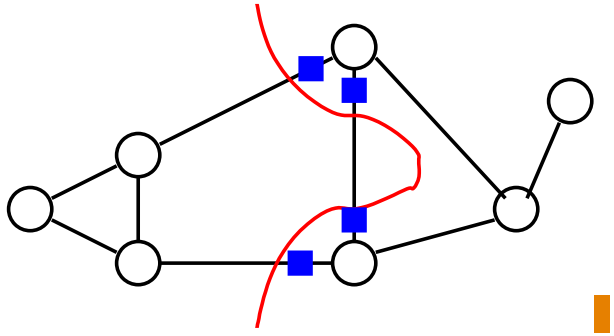
- Eigenschaften der Art: falls Ablauf x, y möglich ist, dann muß auch Ablauf z, y möglich sein.
- Abschlußeigenschaft einer Menge:

$$\sigma \in S \Rightarrow f(\sigma) \subseteq S$$

- Keine Eigenschaft, sondern Eigenschaft einer Eigenschaft, Menge einer Menge von Abläufen.
- *security* = Sicherheit, Lebendigkeit und *noninterference*

security-Komponenten

- Aufteilung des Zustandsraums in Vertraulichkeitszonen, Kontrolle des Informationsflusses auf den Grenzen.



- Protektor für *noninterference* (Abstraktion von Verschlüsselung und Firewall).

security-Theorie

- Formalisierungen:
  - Wissenstheoretische (diskrete) Formalisierung von Vertraulichkeitszonen [Halpern and Moses 1990].
- Mögliches Theorem:
  - Detektoren, Korrektoren, Protektoren hinreichend und notwendig für *noninterference*.

Offene Fragen

- Definition eines “Protektors”?
- Funktionsprinzipien?
- Kompositionsprinzipien?
- Beziehungen zu kryptographischen (komplexitätstheoretischen) Definitionen von *security*?

Security

Beiträge zur Ingenieursmethodik

Einsichten

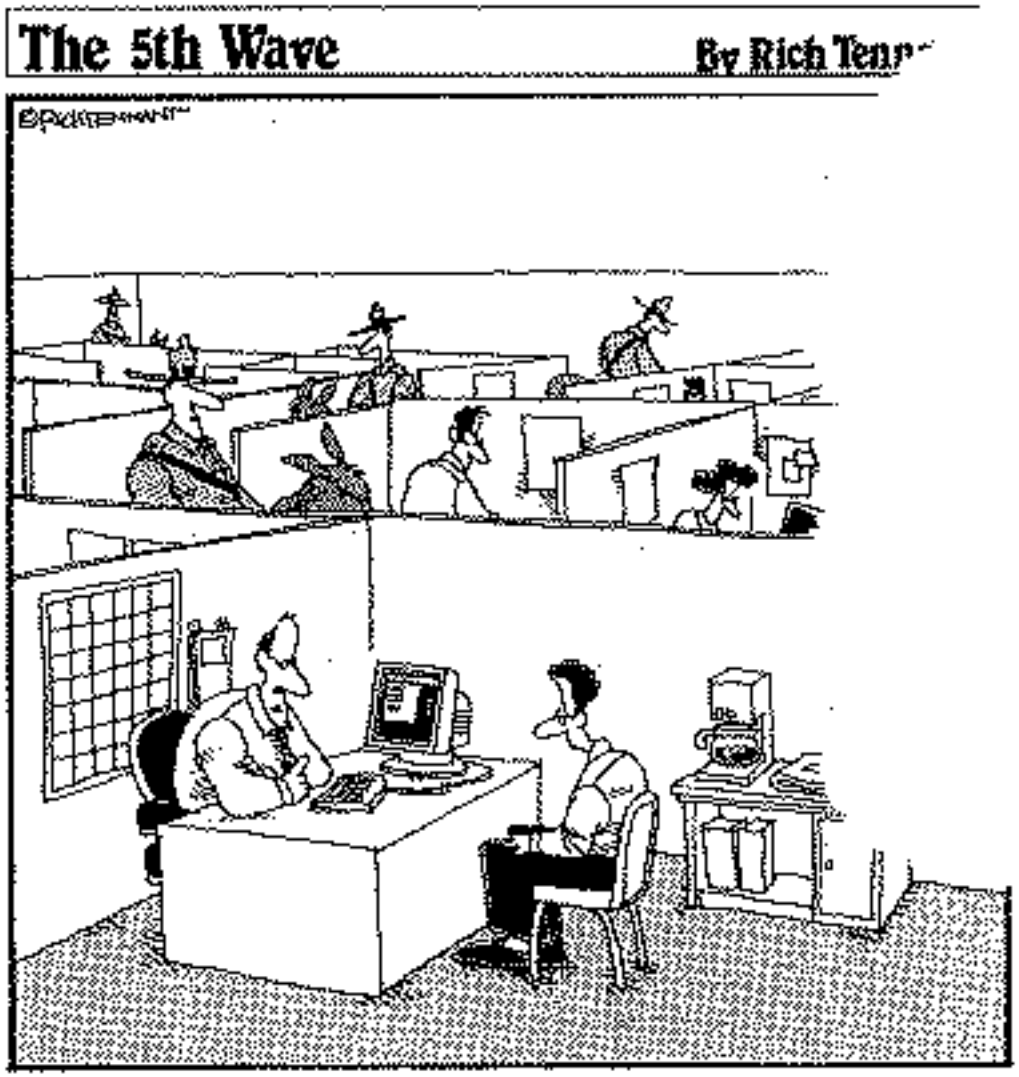
- Sonderstellung von Vertraulichkeit (unerwünschtem Informationsfluß) als Eigenschaft!
- Vertraulichkeit läßt sich (im allgemeinen) nicht verfeinern [McLean 1994]!
- Vertraulichkeit teilweise im Widerspruch zu Sicherheits und Lebendigkeitseigenschaften (Flexibilität vs. Restriktionen).
- Oft: Fehlertoleranzmethodik anwendbar, ähnliche Entwurfsmuster verwenden!

Weitere (gute?) Einsichten

- Aber Fehler $\neq$ Angriffe:
  - Fehler treten zufällig auf, Angriffe nicht.
  - Neue Angriffe verlassen oft das Systemmodell (*timing attacks*, *versioning attacks*), Komplexität des Modells steigt.
  - *Social attacks* = den Menschen mitmodellieren.
- *Security as a process* [Schneier 2000]

Forschungsplan

- Identifizierung von *security*-Komponenten anhand von Fallstudien.
- Einbettung dieser Komponenten in eine allgemeine Theorie (*safety*, *liveness* und *noninterference*).
- Entwurfsmuster und -richtlinien formulieren (Ingenieursintuition).
- Evaluation in der Praxis:
  - 
  - IBM
  - T · · Systems ·


The 5th Wave
By Rich Tenn
„WIR NEHMEN DIE NETZWERK-
SICHERHEIT HIER WIRKLICH ERNST..."

Ausblick (1/2)

- *Security* macht Computersysteme noch komplexer als sie sowieso schon sind.
- Keine einzelne Person weiß, wie und warum auch nur kleine Systeme funktionieren (Reparatur durch Reset, Handauflegen, etc.).
- Akademische Informatik muß ihren Beitrag zu einer abgesicherten aber trotzdem benutzbaren Methodik leisten.

Ausblick (2/2)

- Konflikt zwischen theoretischer Fundierung und Brauchbarkeit in der Praxis nicht unterschätzen!
- These: Gesamtkomplexität wird auf absehbare Zeit nicht beherrschbar werden (*ubiquitous computing*):
  - Müssen mit Risiken leben lernen: Einfache *fallback*-Lösungen.
  - Müssen mit Komplexität leben lernen: Verhaltensstudium von Computersystemen, "Soziologie" der Informatik (z.B. Computerforensik).

Danksagungen

- Folien produziert unter Verwendung von pdfLaTeX und Klaus Guntermanns PPower4.

References

Alpern, B. and Schneider, F. B. 1985. Defining liveness. *Information Processing Letters 21*, 181–185.

Arora, A. and Kulkarni, S. S. 1998. Component based design of multitolerant systems. *IEEE Transactions on Software Engineering 24*, 1 (Jan.), 63–78.

Gärtner, F. C. 1998. Fundamentals of fault tolerant distributed computing in asynchronous environments. Technical Report

TUD-BS-1998-02 (July), Darmstadt University of Technology, Darmstadt, Germany. To appear in *ACM Computing Surveys*, 31(1), March 1999.

Gärtner, F. C. 1999. An exercise in systematically deriving fault-tolerance specifications. In *Proceedings of the Third European Research Seminar on Advances in Distributed Systems (ERSADS)* (Madeira Island, Portugal, April 1999).

Gärtner, F. C. and Völzer, H. 2001. Defining redundancy in fault-tolerant computing. In *Brief Announcement at the 15th International Symposium on DIStributed Computing (DISC 2001)* (Lisbon, Portugal, Oct. 2001).

Gray, III., J. W. and McLean, J. 1995. Using temporal logic to specify and verify cryptographic protocols. In *Proceedings of the Eighth Computer Security Foundations Workshop (CSFW '95)* (Washington - Brussels - Tokyo, June 1995), pp. 108–117. IEEE.

Halpern, J. Y. and Moses, Y. 1990. Knowledge and common knowledge in a distributed environment. *Journal of the ACM 37*, 3 (July),

549–587.

Laprie, J.-C. Ed. 1992. *Dependability: Basic concepts and Terminology*, Volume 5 of *Dependable Computing and Fault-Tolerant Systems*. Springer-Verlag.

McLean, J. 1994. A general theory of composition for trace sets closed under selective interleaving functions. In *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy* (Oakland, CA, 1994), pp. 79–93.

Neumann, P. G. 1995. *Computer Related Risks*. ACM Press.

Schneider, F. B. 1998. Enforceable security policies. Technical Report TR98-1664 (Jan.), Cornell University, Department of Computer Science, Ithaca, New York.

Schneier, B. 2000. *Secrets and Lies: Digital Security in a Networked World*. John Wiley and Sons, Inc., New York, NY, USA.

Zusammenfassung

Die Verletzlichkeit unserer Gesellschaft durch unsichere und unzuverlässige Computersysteme wird allseits beschworen. Dennoch rechtfertigt die Flexibilität, die man durch die Einführung von Computern auch in sensiblen Bereichen der Gesellschaft erreicht, oft die damit einhergehenden Risiken. Schlüssig wird diese Denkweise aber erst, wenn die verwendeten Computer verläßlich sind, d.h. wenn man sich berechtigterweise auf das korrekte Funktionieren des Systems verlassen kann. Die Erfahrung zeigt, dass verläßliche Systeme in vielen Bereichen noch ein Wunschtraum sind. In diesem Vortrag wird die These vertreten, daß nur eine wissenschaftlich fundierte Ingenieursmethodik die steigenden Verläßlichkeitsanforderungen an immer komplexer werdende Rechnersysteme technisch umzusetzen vermag. Der Widerspruch besteht hierbei zwischen der theoretischen Fundierung und der praktischen Umsetzbarkeit. In diesem Vortrag werden einige Beiträge zu einer solchen Methodik vorgestellt, wobei insbesondere auf die Bereiche Fehlertoleranz und Systemsicherheit (im Sinne von *security*) eingegangen wird.