Verteilte Systeme

Oktober - November 2023

9. Vorlesung – 15.11.2023

Kurs: TINF21AI1

Dozent: Tobias Schmitt, M.Eng.

Kontakt: d228143@
student.dhbw-mannheim.de

Wiederholungsfragen

- Welche Umgangsformen hinsichtlich Fehlern sind denkbar?
- Welche Ausfallarten sind denkbar?
- Was verstehen Sie unter den Begriffen: Ausfall, Fehler, Störung?
- Was war noch einmal das Problem der byzantinischen Generäle?
- Was verstehen Sie unter Prozess-Resilienz?
- Welche Möglichkeiten kennen Sie für die Realisierung von Prozess-Resilienz?
- Welche Fehlerklassen können Sie im Rahmen der RPC-Semantik unterscheiden?

Themenüberblick

- **Fehlertoleranz**
  - **Gruppenkommunikation**
  - Verteilter Commit
  - Wiederherstellung (Recovery)
- Sicherheit

Gruppenkommunikation

- Zielsetzung: Zuverlässiges Multicasting
- Gründe:
  - Einzelne Punkt-zu-Punkt-Kommunikation vergeudet Bandbreite
- Probleme:
  - Ein Prozess stürzt während der Kommunikation ab.
  - Ein Prozess kommt während der Kommunikation hinzu.
- Zuverlässigkeit, wenn garantiert werden kann, dass alle nicht fehlerhaften Gruppenmitglieder die Nachricht erhalten.
- Fragestellung:
  - Wer gehört alles zur Gruppe?
  - Wie sieht es hinsichtlich der Reihenfolge aus?

Zuverlässiges Multicasting

- Überlegen Sie sich Möglichkeiten um zuverlässiges Multicasting zu realisieren.

- Siehe z.B. Buch „Verteilte Systeme: Prinzipien und Paradigmen“ von Andrew S. Tanenbaum und Maarten van Steen
(2te aktualisierte Auflage) – Seite 376 bis 380

Zuverlässiges Multicasting

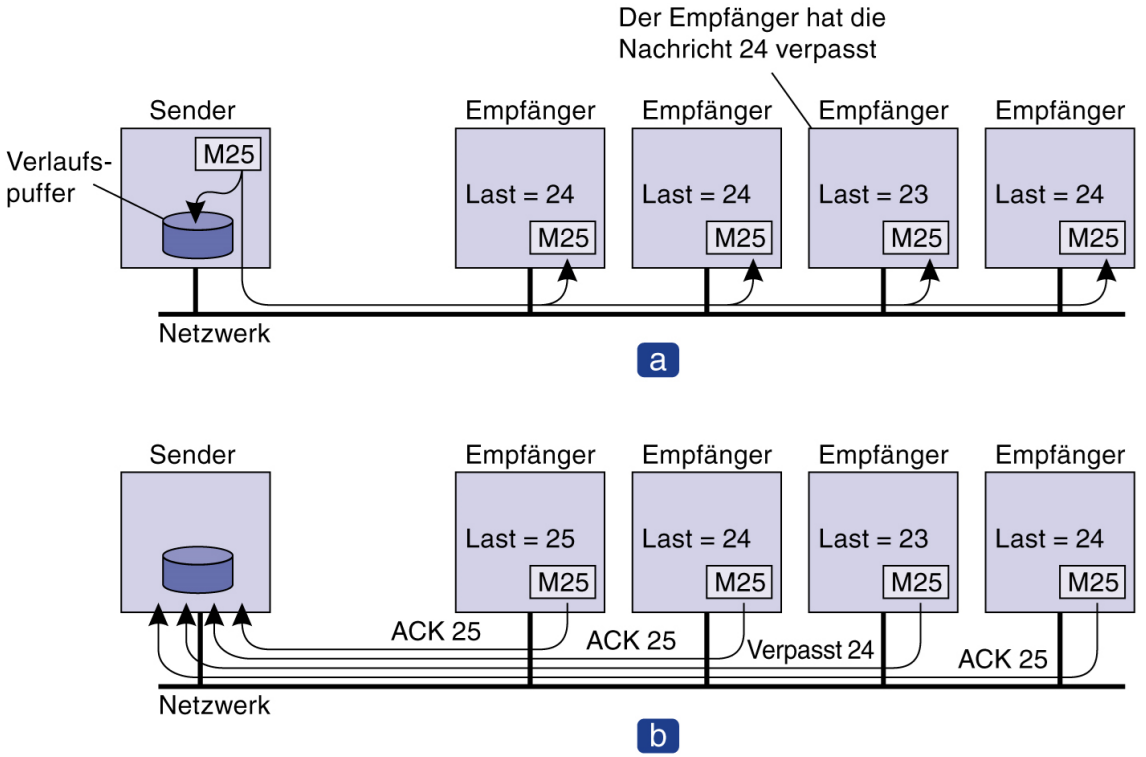
- Realisierung einer schwachen Form des zuverlässigen Multicastings
  - Annahme:
    - 1 Sender und endliche Anzahl an Empfängern
    - zugrundeliegendes Kommunikationssystem unterstützt nur unzuverlässiges Multicasting
      - Nachricht ggf. nicht an alle Gruppenmitglieder aufgrund des Verlustes einer Nachricht

Zuverlässiges Multicasting

- Realisierung einer schwachen Form des zuverlässigen Multicastings
  - Ansatz:
    - Sender weist jeder Nachricht Folgenummer zu
    - Empfänger bestätigen Erhalt
    - Empfänger geben ggf. Rückmeldung, wenn Nachricht fehlt
    - Bedingung: Sender speichert Nachricht im Verlaufspuffer bis alle Empfänger Bestätigung gesendet haben
  - Modifikation:
    - Erneutes Senden der Nachricht, wenn nicht Rückmeldung von allen innerhalb gewisser Zeitspanne
    - Bestätigungsnachrichten im Zusammenhang mit anderen Nachrichten

Zuverlässiges Multicasting

- Realisierung einer schwachen Form des zuverlässigen Multicastings – Teil 2
  - Visualisierung:



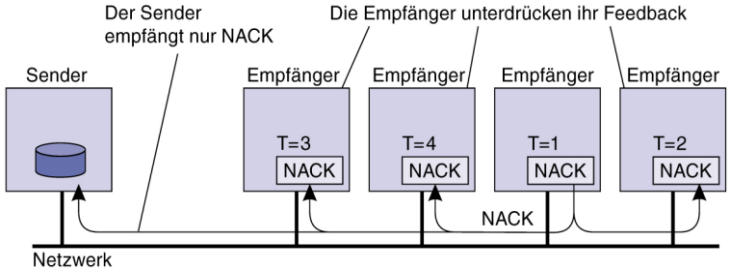
  - Probleme:
    - Rückmeldungsimplosion
      → Skalierbarkeit
    - Ggf. Nachrichten ewig im Verlaufspuffer des Senders

Zuverlässiges Multicasting

- SRM-Protokoll (Scalable Reliable Multicasting)
  - Nachteil
    - Last auf alle Empfänger, die Nachricht erhalten haben
  - Modifikationen
    - Einschränkung des Multicastings durch Untergruppenbildung
    - Empfänger, dem erfolgreich Nachricht zugesandt wurde, kann bei Rückmeldung Nachricht selber via Multicast senden

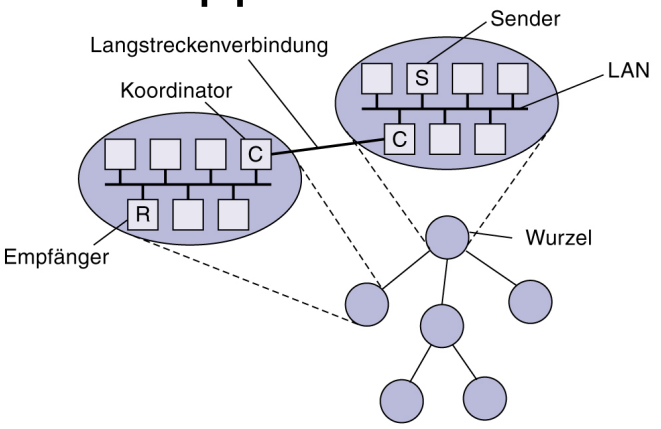
Zuverlässiges Multicasting

- SRM-Protokoll (Scalable Reliable Multicasting)
  - Kernidee: Rückmeldungsunterdrückung
  - Konzept einer nichthierarchischen Rückkopplungssteuerung
  - Realisierung:
    - Erkennen einer fehlenden Nachricht (Achtung: anwendungsabhängig)
    - Rückmelder warten eine zufällige Zeit
    - Rückmeldung via Multicast
    - Keine Rückmeldung nötig, wenn anderer Prozess bereits Rückmeldung gesendet hat



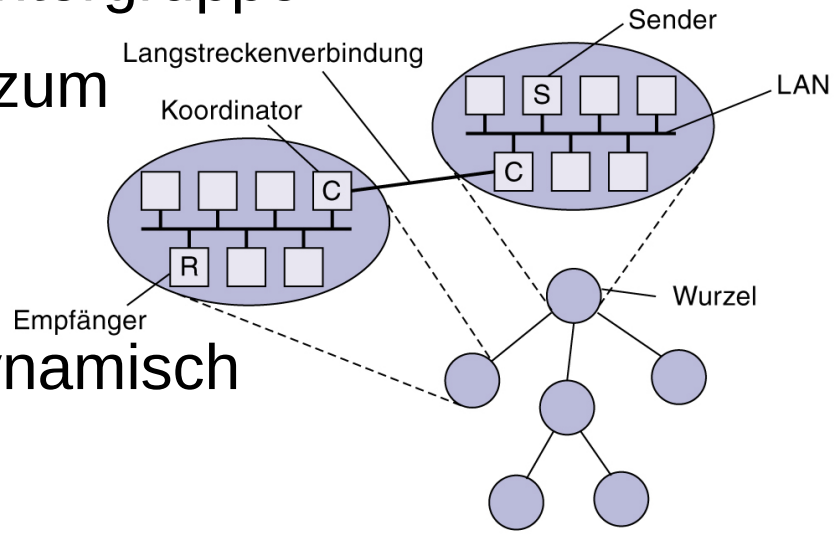
Zuverlässiges Multicasting

- Hierarchische Rückkopplungssteuerung
  - Annahme: 1 Sender und sehr große Gruppe von Empfängern
  - Multicast-Baum
    - Gruppe von Empfängern in Untergruppen unterteilt
    - Anordnung der Untergruppen als Baum
    - Wurzel des Baumes: Untergruppe mit dem Sender
    - Innerhalb jeder Gruppe: beliebiges Verfahren für zuverlässiges Multicasting für kleine Gruppen



Zuverlässiges Multicasting

- Hierarchische Rückkopplungssteuerung
  - Jede Gruppe hat einen Koordinator
  - Koordinator meldet Erhalt bzw. Fehlen von Nachrichten an übergeordnete Untergruppe
  - Koordinator bearbeitet Anfragen zum erneuten Senden
  - Problematik:
    - Aufbau des Baumes, da meist dynamisch

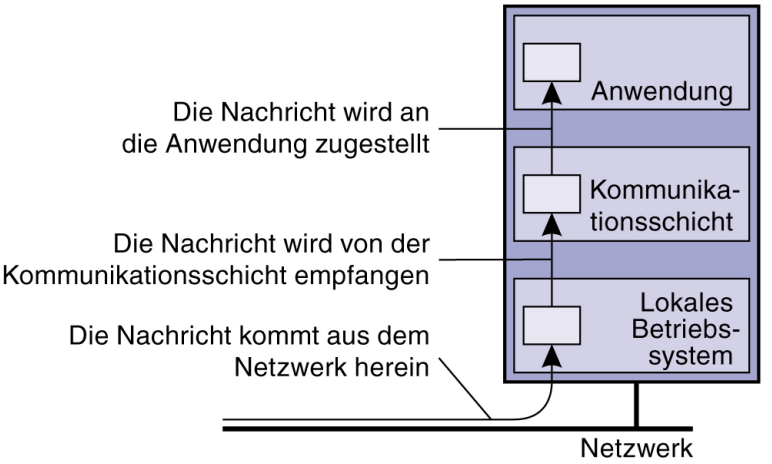


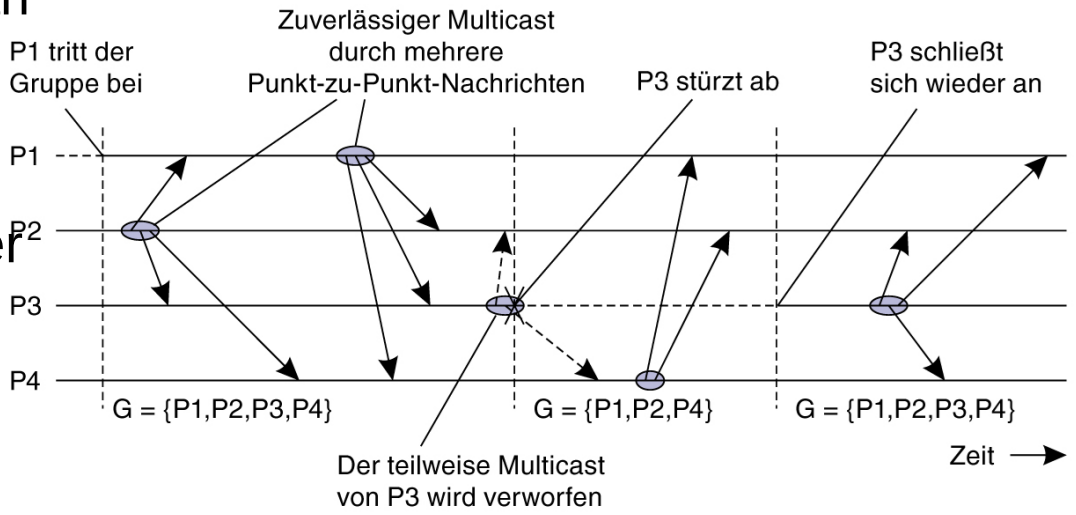
Atomares Multicasting

- Begrifflichkeit Atomares Multicasting
  - Zuverlässiges Multicasting bei Vorliegen von Prozessausfällen
  - Garantie: Auslieferung an alle oder keinen
  - Alle Nachrichten in der gleichen Reihenfolge
- Konzept
  - Klare Entscheidungen bzgl. Gruppenmitgliedschaft
    - Aktualisierung erst, wenn abgestürzte Replik nicht zur Gruppe
    - Neue bzw. rebootete Repliken erhalten erst Aktualisierungen, wenn sie als Mitglied registriert sind
  - Beitritt zu einer Gruppe nur wenn der gleiche Zustand wie alle anderen Gruppenmitglieder
  - Nicht fehlerhafte Prozesse haben konsistente Sicht auf Datengrundlage

Atomares Multicasting
Virtuelle Gleichzeitigkeit

- Realisierung atomaren Multicastings – Architekturkonzepte
  - Unterscheidung zwischen Empfang und Auslieferung
  - Einführung der Gruppensicht
    - Änderung der Gruppensicht meint einen Ein- oder Austritt eines Prozess zu der Gruppe
  - Virtuelle Gleichzeitigkeit:
    - Problem: Während Senden an Gruppe stürzt Sender ab
    - Resultatmöglichkeiten
      - Dennoch Auslieferung der Nachricht an alle übrigen Prozesse
      - Nachricht wird ignoriert





Atomares Multicasting
Anordnung von Nachrichten

- Unterscheidung von Multicasts
  - Nicht geordnete Multicasts
  - FIFO-geordnete Multicasts
  - Kausal geordnete Multicasts
  - Total geordnete Multicasts

- Nicht geordnete Multicasts
  - Keine Garantien bzgl. Anordnung

Prozess P1	Prozess P2	Prozess P3
Sendet m1	Empfängt m1	Empfängt m2
Sendet m2	Empfängt m2	Empfängt m1

- FIFO-geordnete Multicasts
  - Eingehende Nachrichten des gleichen Prozesses überall in der gleichen Reihenfolge

Prozess P1	Prozess P2	Prozess P3	Prozess P4
Sendet m1	Empfängt m1	Empfängt m3	Sendet m3
Sendet m2	Empfängt m3	Empfängt m1	Sendet m4
	Empfängt m2	Empfängt m2	
	Empfängt m4	Empfängt m4	

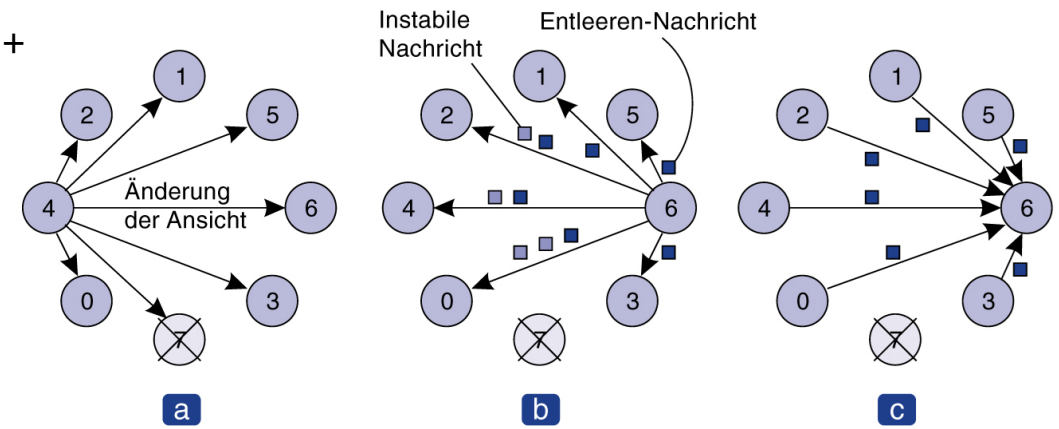
Atomares Multicasting
Anordnung von Nachrichten

- Kausal geordnete Multicasts
  - Kausalität zwischen unterschiedlichen Nachrichten bleibt erhalten (Nachrichten ggf. von unterschiedlichen Sendern)
- Total geordnete Multicasts
  - Alle Gruppenmitglieder sehen dieselbe Reihenfolge
  - Keine Angabe über Art der Ordnung

- Atomares Multicasting meint
Virtuell gleichzeitiges zuverlässiges Multicasting mit total geordneter Auslieferung
  - Auslieferung via FIFO oder kausal möglich

Atomares Multicasting

- Idee einer Realisierung virtueller Gleichzeitigkeit
  - Grunddesign:
    - Verwendung zuverlässiger Punkt-zu-Punkt-Kommunikation (TCP)
    - → Übertragung garantiert erfolgreich,
    - aber Senderabsturz bevor alle Gruppenmitglieder etwas erhalten haben
  - Vorgehen bzgl. Nachrichten:
    - Prozess in G speichert Nachrichten
    - Auslieferung erst, wenn Nachricht stabil = Nachricht von alle Prozesse von G empfangen
    - Sicherstellung der Stabilität – beliebiger (funktionierender) Prozess sendet Nachricht noch einmal an alle anderen Prozesse
  - Vorgehen bei Änderung der Gruppensicht:
    - (a) Erhalt einer Nachricht für Wechsel auf G_{i+1}
    - (b) Senden aller instabilen Nachrichten +
    - Senden einer Flush-Nachricht (Leerungsnachricht)
    - (c) Wechsel auf G_{i+1} bei Erhalt aller andere Flush-Nachrichten



Themenüberblick

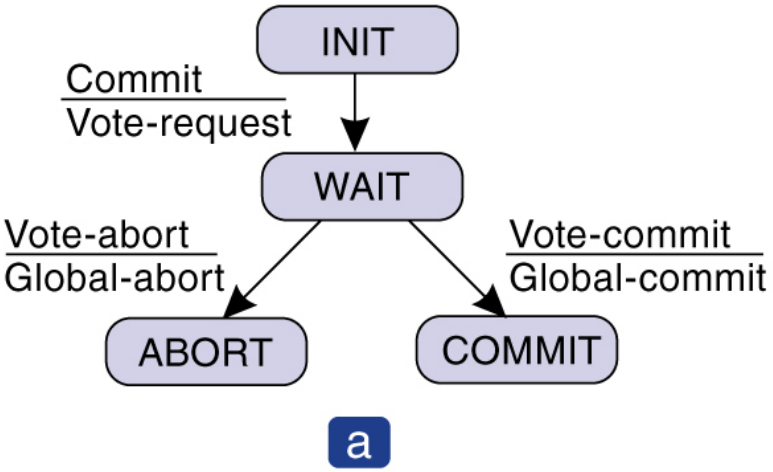
- **Fehlertoleranz**
  - Gruppenkommunikation
  - **Verteilter Commit**
  - Wiederherstellung (Recovery)
- Sicherheit

Verteilter Commit

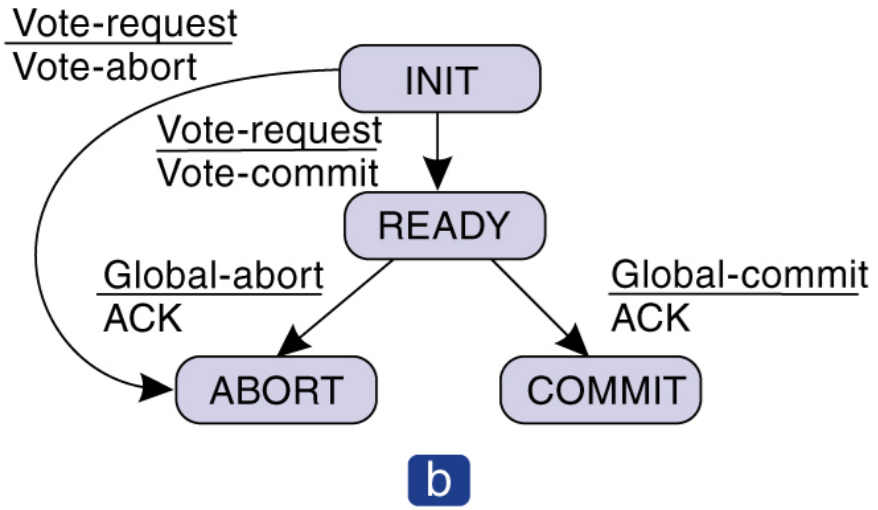
- Beispiele für verteilten Commit
  - Auslieferung einer Nachricht
  - Festschreiben einer Transaktion
  - …

- Ein-Phasen-Commit-Protokoll
  - Koordinator teilt allen beteiligten Prozessen mit, dass Operation (lokal) ausgeführt werden soll
  - Keine Rückmeldung der anderen Prozesse, falls Ausführung nicht möglich

Zwei-Phasen-Commit (2PC)



Koordinator

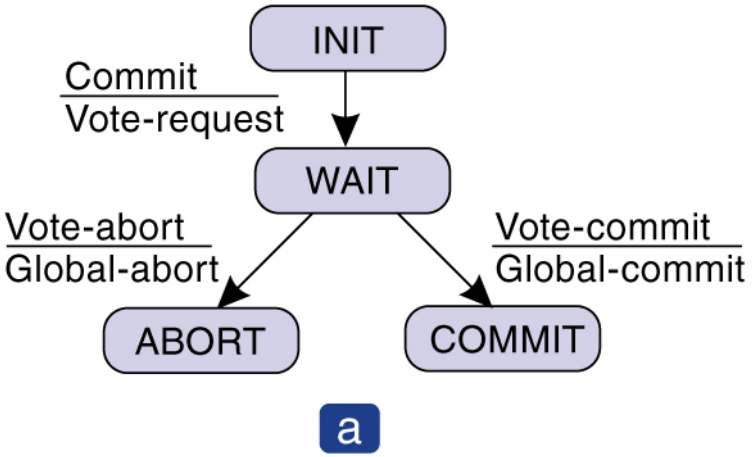


Teilnehmer

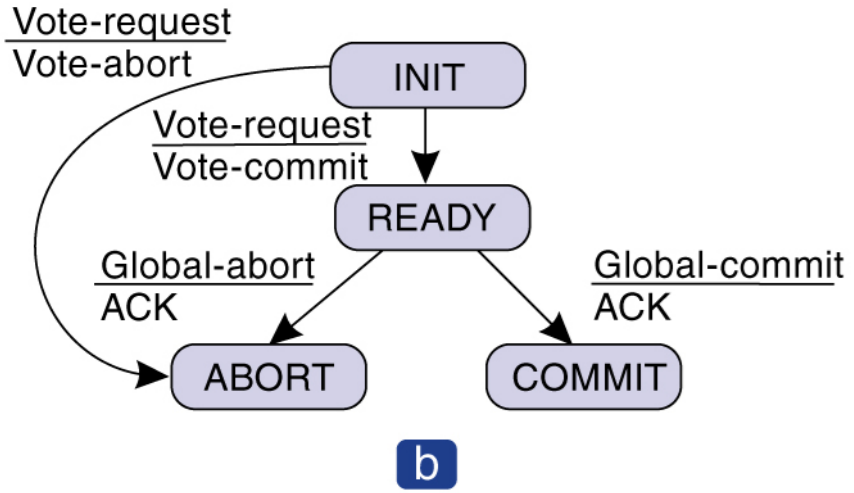
Zwei-Phasen-Commit (2PC)

- Abstimmungsphase
  - Koordinator sendet VOTE_REQUEST
  - Empfänger senden VOTE_COMMIT (Ausführung mgl.) oder VOTE_ABORT
- Entscheidungsphase
  - Koordinator sammelt alle Rückmeldungen
  - Koordinator sendet GLOBAL_COMMIT, wenn alle Teilnehmer VOTE_COMMIT gesendet haben oder andernfalls GLOBAL_ABORT
  - Sender von VOTE_COMMIT warten auf Bestätigung für Entgültiges Festschreiben
    - bei GLOBAL_COMMIT → Entgültiges Festschreiben
    - bei GLOBAL_ABORT → Abruch

Zwei-Phasen-Commit (2PC)



Koordinator



Teilnehmer

Zwei-Phasen-Commit (2PC)

- Umgangsproblematik – Absturz des Koordinators oder eines Teilnehmers
  - Umgang hinsichtlich unendlich langem Blockieren (bei fehlenden Rückmeldungen)
  - Umgang hinsichtlich Wiederherstellung

- Umgangsform – Blockieren
  - Blockieren eines Teilnehmers im INIT-Zustand
    - VOTE_REQUEST wird nicht empfangen
    - Zeitintervall warten, dann Senden eines VOTE_ABORT

Zwei-Phasen-Commit (2PC)

- Umgangsform – Blockieren
  - Blockieren eines Teilnehmers im INIT-Zustand
    - VOTE_REQUEST wird nicht empfangen
    - Zeitintervall warten, dann Senden eines VOTE_ABORT
  - Blockieren des Koordinator im WAIT-Zustand
    - Fehlen von VOTE_COMMIT-Nachrichten
    - Zeitintervall warten, dann Senden eines GLOBAL_ABORT
  - Blockieren eines Teilnehmers im READY-Zustand
    - Fehlen der GLOBAL_COMMIT- oder GLOBAL_ABORT-Nachricht

Zwei-Phasen-Commit (2PC)

- Umgangsform – Blockieren
  - Blockieren eines Teilnehmers im READY-Zustand
    - Fehlen der GLOBAL_COMMIT- oder GLOBAL_ABORT-Nachricht
    - Anderen Teilnehmer kontaktieren
      - Anderer Teilnehmer erhielt GLOBAL_X → Ausführen von GLOBAL_X
      - Anderer Teilnehmer noch im INIT-Zustand → Senden von VOTE_ABORT
      - Alle anderen Teilnehmer im READY-Zustand → Warten bis Wiederherstellung des Koordinators

Zwei-Phasen-Commit (2PC)

- Fehlerquelle – Absturz des Koordinators oder eines Teilnehmers
  - Ziel: Wiederherstellung
  - Lösungsansatz: Festschreiben des Zustandes im nicht flüchtigen Speicher
    - Prinzip des Journaling
  - Koordinator schreibt folgende Zustände in lokale Protokolldatei:
    - Start des Zwei-Phasen-Commits
    - GLOBAL_ABORT oder GLOBAL_COMMIT

Zwei-Phasen-Commit (2PC)

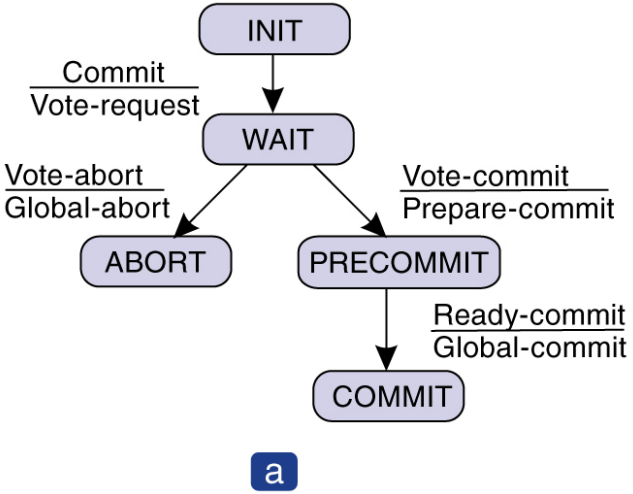
- Fehlerquelle – Absturz des Koordinators oder eines Teilnehmers
  - Teilnehmer schreibt folgende Zustände in lokale Protokolldatei:
    - INIT, VOTE_X, GLOBAL_X
    - DECISION_REQUEST (Anfrage via Multicast hinsichtlich Entscheidung des Koordinators an andere Teilnehmer)

- Hinweis:
  - 2PC ist ein blockierendes Commit-Protokoll, da Absturz des Koordinators vor Senden von GLOBAL_X alle blockiert

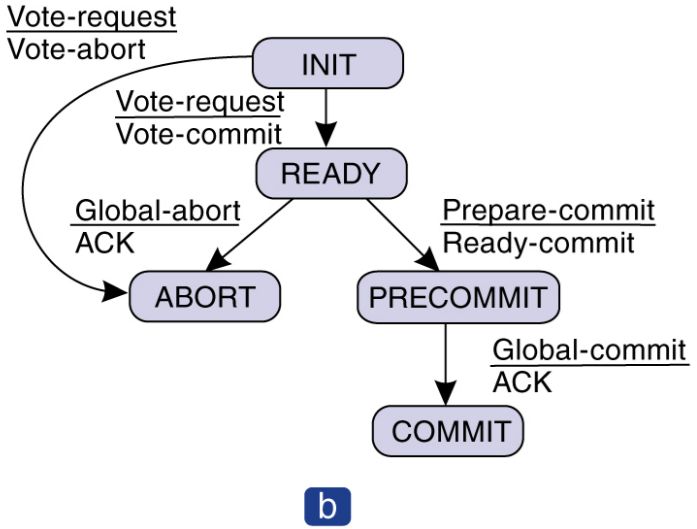
Drei-Phasen-Commit (3PC)

- Erweiterung von 2PC - nicht blockierendes Commit-Protokoll
- Maximen hinsichtlich des Zustandes des Koordinators und aller Teilnehmer
  - *Es gibt keinen einzelnen Zustand, von dem aus es möglich ist, direkt in einen COMMIT- oder ABORT-Zustand überzugehen.*
  - *Es gibt keinen Zustand, in dem es nicht möglich ist, eine endgültige Entscheidung zu treffen, und von dem aus in einen COMMIT-Zustand übergegangen werden kann.*
- Darstellung als endliche Zustandsautomaten:

Drei-Phasen-Commit (3PC)

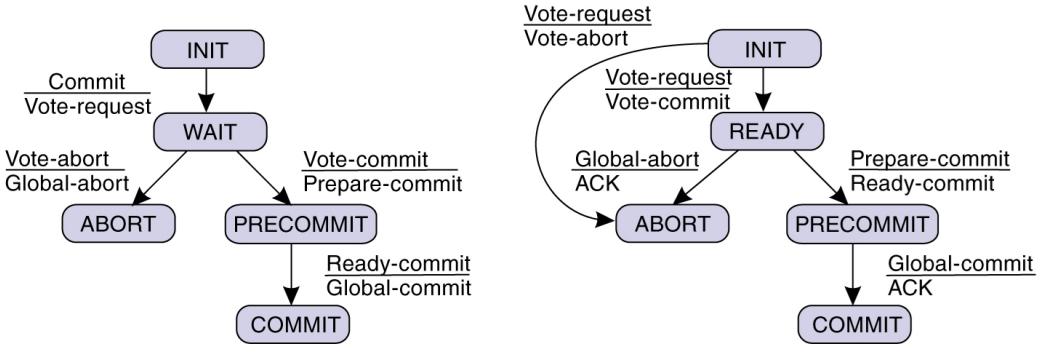


Koordinator



Teilnehmer

Drei-Phasen-Commit (3PC)



Koordinator Teilnehmer

- Was passiert, wenn alle Teilnehmer im Zustand PRECOMMIT blockieren?

- Was passiert, wenn sich der Koordinator im Zustand WAIT befindet und nicht alle VOTE_COMMIT-Nachrichten eintreffen?

- Kann sich ein Teilnehmer im Zustand INIT befinden, während sich ein anderer Teilnehmer im Zustand PRECOMMIT befindet?

Themenüberblick

- **Fehlertoleranz**
  - Gruppenkommunikation
  - Verteilter Commit
  - **Wiederherstellung (Recovery)**
- Sicherheit

Wiederherstellung (Recovery)

- Grundgedanke: Wiederherstellung nach einem Fehler
  - Ersetzung eines fehlerhaften Zustandes durch einen fehlerfreien Zustand
- Ansätze
  - Vorwärtswiederherstellung (Forward Recovery)
    - Versucht das System in ein Zustand zu bringen, in dem das System weiterlaufen kann

Wiederherstellung (Recovery)

- Rückwärtswiederherstellung (Backward Recovery)
  - Zurücksetzung des System zu einem fehlerfreien vorangegangen Zustand
  - Notwendigkeit von Kontrollpunkten (Checkpoints), Aufzeichnung des Zustandes des gesamten Systems (oder eines Teils)
  - Übliche Form der Wiederhestellung

- Achtung: Rückwärtswiederherstellung nicht immer möglich (bei Vorliegen irreversible Operationen)
  - Fehlerhafter Bankautomat, der einfach 1000 € ausgibt
  - Kommando **rm -rf ***
  - **...**

Wiederherstellung (Recovery)

- Erstellung von Kontrollpunkten ist aufwendig!

- Erweiterung hinsichtlich Rückwärtswiederherstellung
  - Protokollierung der Nachrichten nach einem Checkpoint (Message Logging)
  - Effizienter als reines Arbeiten mit Kontrollpunkten
  - Umgangsformen
    - Senderbasierte Protokollierung
    - Empfängerbasierte Protokollierung

- Hinweis
  - Wiederherstellung in verteilten Systemen schwierig
  - Zusammenarbeit zwischen Prozessen notwendig, um einen konsistenten wiederherzustellenden Zustand zu finden

Interludium: Stabiler Speicher

- Arbeit mit Kontrollpunkten verlangt sicheres Speichern der Daten
- Infos müssen
  - Prozess- und Systemabstürze überleben
  - Ausfälle des Speichermediums überleben
- Einfacher Ansatz für stabilen Speicher

Interludium: Stabiler Speicher

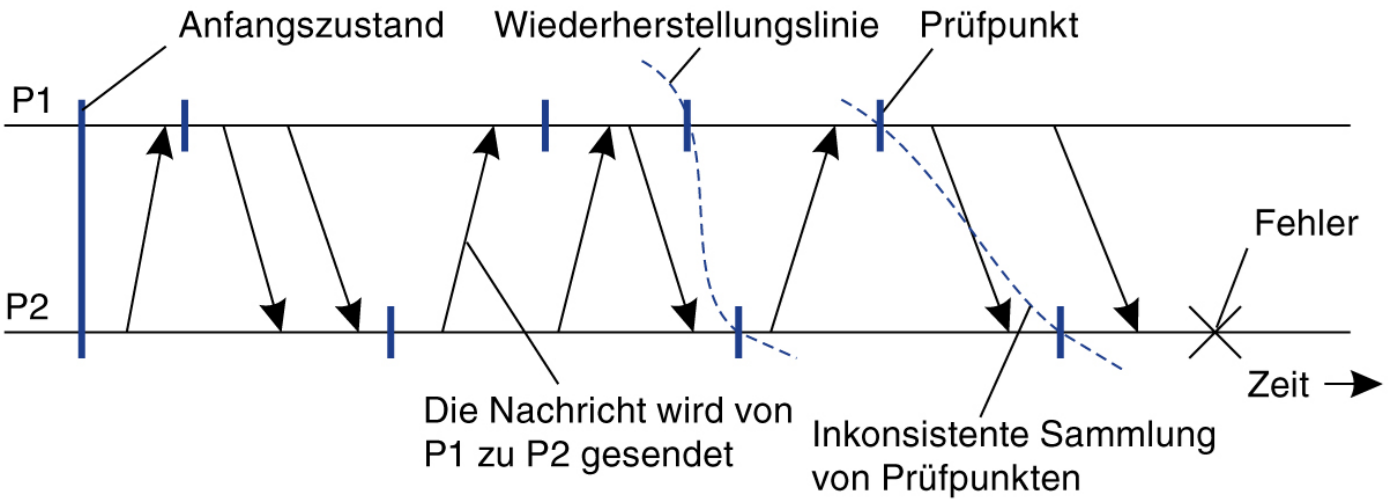
- Arbeit mit 2 Festplatten, Verwendung von Prüfsummen, Einsatz von Journaling
- Arbeitsweise:
  - Schreibe auf Festplatte 1, dann auf Festplatte 2
- Fehlerbehandlung:
  - Prüfsumme passt, aber Infos auf 1 abweichend von Infos auf 2
    → Übernahme der Infos von Festplatte 1
  - Prüfsummer nicht richtig
    → Übernahme der Infos von der anderen Festplatte

- Alternativer Ansatz für stabilen Speicher:
  Entsprechend konfiguriertes RAID-System

Kontrollpunkt (Checkpoint)

- Prinzipien für Fehlertoleranz
  - Regelmäßiges Speichern in stabilen Speicher
  - Aufzeichnung eines konsistenten globalen Zustandes
    Bezeichnung: Verteilte Momentaufnahme (Distributed Snapshot)
- Schwierigkeiten im Umgang mit Nachrichten
  - Prozess P1 - Empfang einer Nachricht ist aufgezeichnet
  - Prozess P2 - Existenz der Aufzeichnung hinsichtlich des Sendens
- Ziel: Wiederherstellung der letzten verteilten Momentaufnahme (letzte konsistente Erfassung von Kontrollpunkten)
  - Finden der Wiederherstellungslinie (Recovery Line)
- Beispiel:

Kontrollpunkt (Checkpoint)

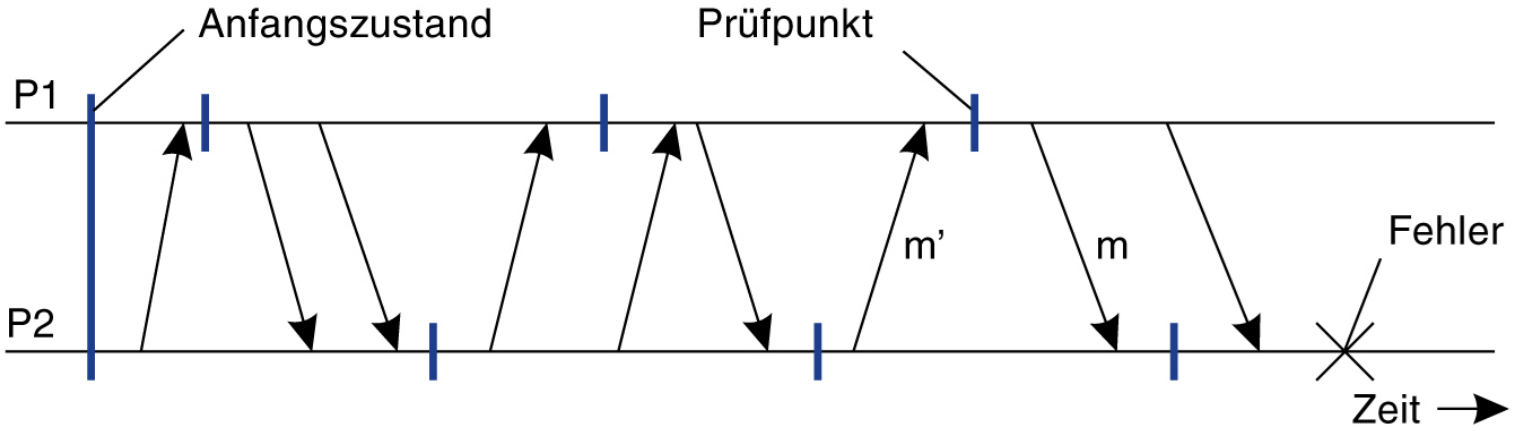


Kontrollpunkt (Checkpoint)

- Wie kann das Setzen von Kontrollpunkten in einem verteilten System koordiniert werden?

- Welche Bedingungen müssen auf den einzelnen Systemen vorliegen, damit die Kontrollpunkte auch fehlertolerant gespeichert werden? Wo speichert man Kontrollpunkte am besten?

- Finden Sie in der Abbildung die Wiederherstellungslinie!



Kontrollpunkt (Checkpoint)

- Arbeitsweisen
  - Unabhängige Kontrollpunkte (Independent Checkpointing)
    - Berechnung des letzten konsistenten Erfassung von Kontrollpunkten schwierig / aufwendig
    - Dominoeffekt
      - Kaskadierendes Zurückrollen
      - Worst Case: Zurückrollen auf den Anfangszustand des Systems
    - Regelmäßige Speicherbereinigung notwendig
  - Koordinierte Kontrollpunkte (Coordinated Checkpointing)
    - …

Kontrollpunkt (Checkpoint)

- Koordinierte Kontrollpunkte (Coordinated Checkpointing)
  - Notwendigkeit einer globalen Synchronisierung
  - Höhere Leistungsanforderung
  - Abstimmung z.B. durch Zwei-Phasen-Protokoll (2PC)
  - Global konsistenter Zustand wird erreicht
    - Eingehende Nachrichten nach Kontrollpunkt gehören nicht zum Kontrollpunkt
    - Ausgehende Nachrichten kommen in Warteschlange bis Kontrollpunkt fertiggestellt

Kontrollpunkt (Checkpoint)

- Koordinierte Kontrollpunkte (Coordinated Checkpointing)
  - Optimierung
    - Inkrementelle Momentaufnahme (Incremental Snapshot)
    - Schritt 1: An welche Prozesse ging nach dem letzten Kontrollpunkt eine Nachricht?
    - Schritt 2: Nur jene Prozesse erstellen Kontrollpunkt.

Nachrichtenprotokollierung

- Annahme: Stückweise deterministisches Modell
  - Ausführung jedes Prozesses in einer Folge von Intervallen in denen Ereignisse stattfinden
  - Ereignisse sind vollständig geordnet (Anwendung der Passiert-vor-Relation)
  - Beginn des Intervalls mit nichtdeterministischen Ereignis
  - Nach erster Nachricht → System vollständig deterministisch

- Ziel: Protokollierung aller nichtdeterministischen Ereignisse
  - Nutzung von stabilen Nachrichten
    (Stabile Nachricht = Nachricht kann nicht verloren gehen, weil z.B. im stabilen Speicher hinterlegt)

Themenüberblick

- Fehlertoleranz
  - Gruppenkommunikation
  - Verteilter Commit
  - Wiederherstellung (Recovery)
- **Sicherheit**
  - **Grundlagen und Entwurfsfragen**
  - Kryptografie

Sicherheit - Grundlagen

- Systemstabilität
  - Verfügbarkeit, Zuverlässigkeit, Funktionssicherheit, Wartbarkeit
- Weitere Sicherheitsaspekte
  - Vertraulichkeit (Confidentiality)
    - Eigenschaft eines Computersystems Infos nur an autorisierte Parteien preiszugeben
  - Integrität
    - Eigenschaft, dass nur autorisierte Änderungen an Bestandteilen des Systems vorgenommen werden können
- Bestandteile jedes Computersystem
  - Hardware, Software und Daten

Sicherheit - Grundlagen

- Sicherheitsbedrohungen (Security Threads) hinsichtlich gebotener Dienste und Daten
  - Abfangen (Interception)
    - Zugriff auf Dienst oder Daten
    - Bsp.: Abhören einer Kommunikation oder illegales Kopieren von Daten, nach Eindringen in ein privates Benutzerverzeichnis
  - Stören (Interruption)
    - Allgemein: keine Verfügbarkeit des Dienstes oder von Daten
    - Bsp.: gelöschte Datei oder DoS-Angriffe

Sicherheit - Grundlagen

- Sicherheitsbedrohungen (Security Threads) hinsichtlich gebotener Dienste und Daten
  - Verändern (Modification)
    - Unautorisierte Änderungen an Daten (z.B. von Datenbankeinträgen)
    - Manipulation an einem Dienst (z.B.: zur Aufzeichnung der Benutzeraktivitäten)
  - Einbringen (Fabrication) / Fälschung
    - Erzeugung zusätzlicher Daten oder Aktivitäten
    - z.B. neuer Eintrag in eine Passwortdatei oder Wiederholung einer zuvor gesendeten Nachricht

Sicherheit - Grundlagen

- Ziele, Bedrohungen und Maßnahmen hinsichtlich Sicherheit und Datenschutz

- Hauptziel: Unternehmenserfolg
- Formalziele: Rechtmäßigkeit, Wirtschaftlichkeit, Akzeptanz
- Schutzziele: Vertraulichkeit, Integrität, Verfügbarkeit, Verbindlichkeit

- Bedrohungen: Höhere Gewalt, Technisches Versagen, Menschliches Versagen, Spionage, Sabotage, Betrug

- Maßnahmenbereiche bzgl. Sicherheit und Datenschutz: Infrastruktur, Personal, Organisation sowie Hardware, Software, Kommunikationstechnik

(Vgl. https://www.yumpu.com/de/document/read/20669712/zusammenfassung-kapitel-8 Folie 12)

Sicherheit - Grundlagen

- Weg zu einem sicheren System
  - Beschreibung der Sicherheitsanforderungen bzw. der Sicherheitsrichtlinien
    - Detaillierte Beschreibung, welche Aktionen die Entitäten in einem System befugt und welche verboten sind
  - Durchsetzung der Sicherheitsrichtlinien durch Sicherheitsmechanismen

Sicherheit - Grundlagen

- Weg zu einem sicheren System
  - Sicherheitsmechanismen sind
    - Verschlüsselung
      - Mittel zur Umsetzung von Vertraulichkeit und Prüfung hinsichtlich Modifikation
    - Authentifizierung (Identifikation)
    - Autorisierung (Erlaubnis für Ausführungen)
    - Kontrolle
      - Kein Schutz vor Sicherheitsbedrohen, aber hilfreich für die Analyse eines Sicherheitseinbruchs und für Einleitung von Maßnahmen gegen Eindringlinge

Sicherheit - Grundlagen

- Begrifflichkeiten
  - Sicherheit (safety)
    - Nicht technische Herausforderungen
  - Schutz (protection / security)
    - Technische und organisatorische Maßnahmen
    - Bereich der Sicherheits- bzw. Schutzmechanismen
- Hinweis:
  - Man möchte gerne Sicherheit, aber man kann nur Schutzmechanismen anbieten.

Sicherheit - Grundlagen

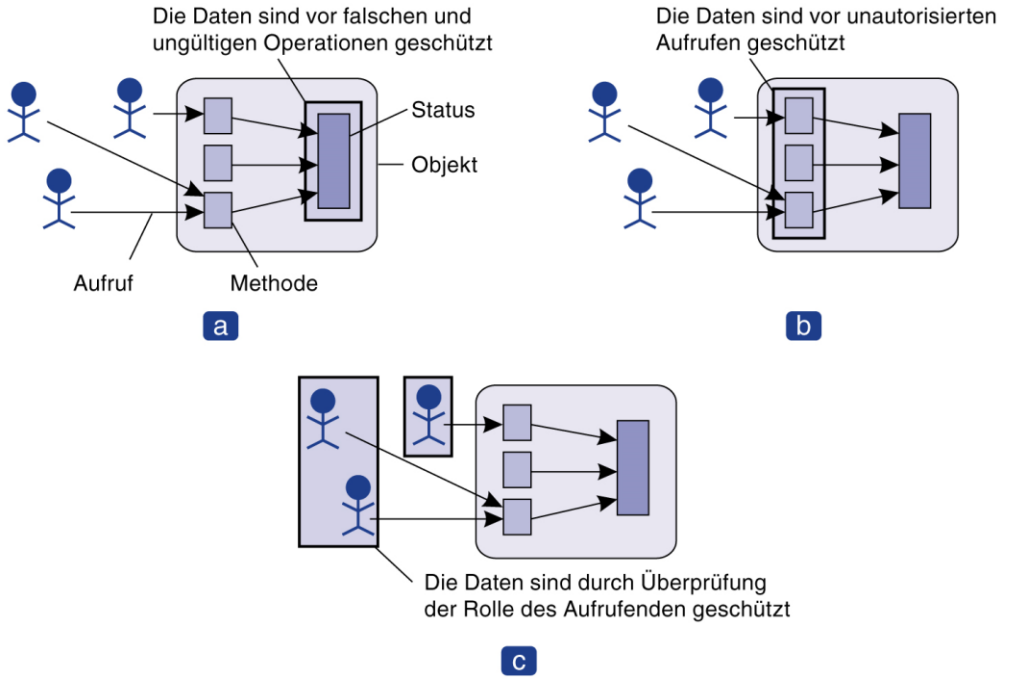
- Sicherheitsmechanismen allgemein
  - Algorithmisch fassbare Lösungsprinzipien
  - Bereitstellung als
    - Spezifizierung der Funktionsweise
    - Implementation
  - Beispiele für Sicherheitsmechanismen
    - Adressraumseparierung
    - Asymmetrische Verschlüsselung
    - Auditing
    - Firewalls
    - Viren-Erkennungsprogramme
    - Zugriffskontrolllisten
    - ...

(Vgl. https://www.yumpu.com/de/document/read/20669712/zusammenfassung-kapitel-8 Folie 17)

Sicherheit - Entwurfsfragen

.**Kontrollfokus** – Ansätze zum Schutz von Sicherheitsbedrohungen

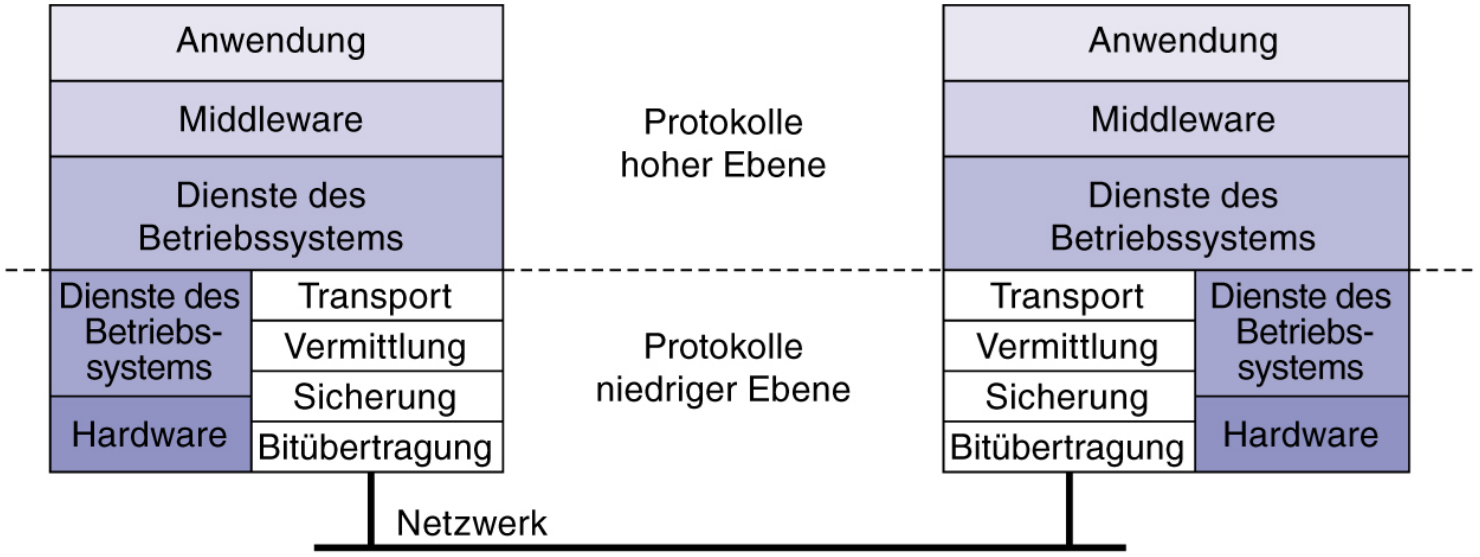
- (a) Schutz vor ungültigen Operationen
- (b) Schutz vor unautorisierten Aufrufen
- (c) Schutz vor unautorisierten Benutzern



Sicherheit - Entwurfsfragen

.Schichten der Sicherheitsmechanismen

- Frage: Auf welcher Ebene müssen Sicherheitsmechanismen platziert werden?
- Hinweis bzgl. verteilter Systeme: Sicherheitsmechanismen meist auf der Schicht der Middleware
- Problematik des Vertrauens in die Sicherheit zugrundeliegender Schichten



Sicherheit - Entwurfsfragen

- **Verteilung von Sicherheitsmechanismen**
  - Bestehen von Abhängigkeiten hinsichtlich des Vertrauens → Trusted Computing Base (TCB)
  - TCB = Menge aller Sicherheitsmechanismen in einem (verteilten) Computersystem, die zur Durchsetzung von Sicherheitsrichtlinien benötigt werden … und denen deshalb vertraut werden muss.

Sicherheit - Entwurfsfragen

- **Verteilung von Sicherheitsmechanismen**
  - Verteiltes System → Sicherheit beruht auf dem zugrunde liegenden lokalen Betriebssystem.
    - Verwendung von VMs → Vertrauen beruht auf dem verwendeten Hypervisor
      (vgl. https://www.theregister.com/2020/03/17/virtual_machines_patch/)
  - Konsequenz: Trennung von Sicherheitssystemen von anderen Diensten durch Nutzung verschiedener Computer mit entsprechendem Grad an Sicherheit
    - z.B. Dateiserver auf Rechner mit vertrauenswürdigem Betriebssystem
      und Ausführung der Clients auf „unsicheren“ Computern

Sicherheit - Entwurfsfragen

- **Einfachheit**
  - „Einfachheit trägt zum Vertrauen bei, dass Endbenutzer in die Anwendung setzen, und, was wichtiger ist, zu der Überzeugung der Entwickler, dass das System keine Sicherheitslücken hat.“
  - Prinzip: „keep it simple“ → Zusätzliche Features können mehr Angriffspunkte liefern.
    - z.B. E-Mails in Plain-Text sicherer
    - z.B. E-Mail-Programm mit automatischen Öffnen der Anhänge unsicherer

Sicherheit - Entwurfsfragen

- **Einfachheit**
  - „Verteiltes System → mitunter hohe Komplexität
    - Anwendung von relativ einfachen und leicht verständlichen Mechanismen
  - Beispiel: Arbeit mit Mircoservices
    - Unabhängigen Prozessen kommunizieren über API / Programmierschnittstelle
    - Microservice sollte von jedem Teammitglied überschaubar sein und in vertretbarem Zeitaufwand erstellt werden
    - Siehe https://de.wikipedia.org/wiki/Microservices

Themenüberblick

- Fehlertoleranz
  - Gruppenkommunikation
  - Verteilter Commit
  - Wiederherstellung (Recovery)
- **Sicherheit**
  - Grundlagen und Entwurfsfragen
  - **Kryptografie**

Kryptographie - Einstiegsfragen

- Kinder überlegen sich für jeden Buchstaben ein neues Zeichen. Ist die Verschlüsselung sicher?

- Daten werden in einem binären Format in einer Datei abgelegt. Vorausgesetzt, dass das Datenformat nicht veröffentlicht wurde, sind dann die Daten sicher?

- Eine Softwareschmiede bietet ein High-End-Verschlüsselungsprodukt an. Aus Gründen des Marktvorteils will die Softwareschmiede den Algorithmus nicht veröffentlichen. Wie sicher sind dann Daten, die mit der Software verschlüsselt wurden?

- Ab wann gilt ein Verschlüsselungsalgorithmus als gebrochen?

- Worin liegt die Effizienz eines Kyptographie-Verfahrens?

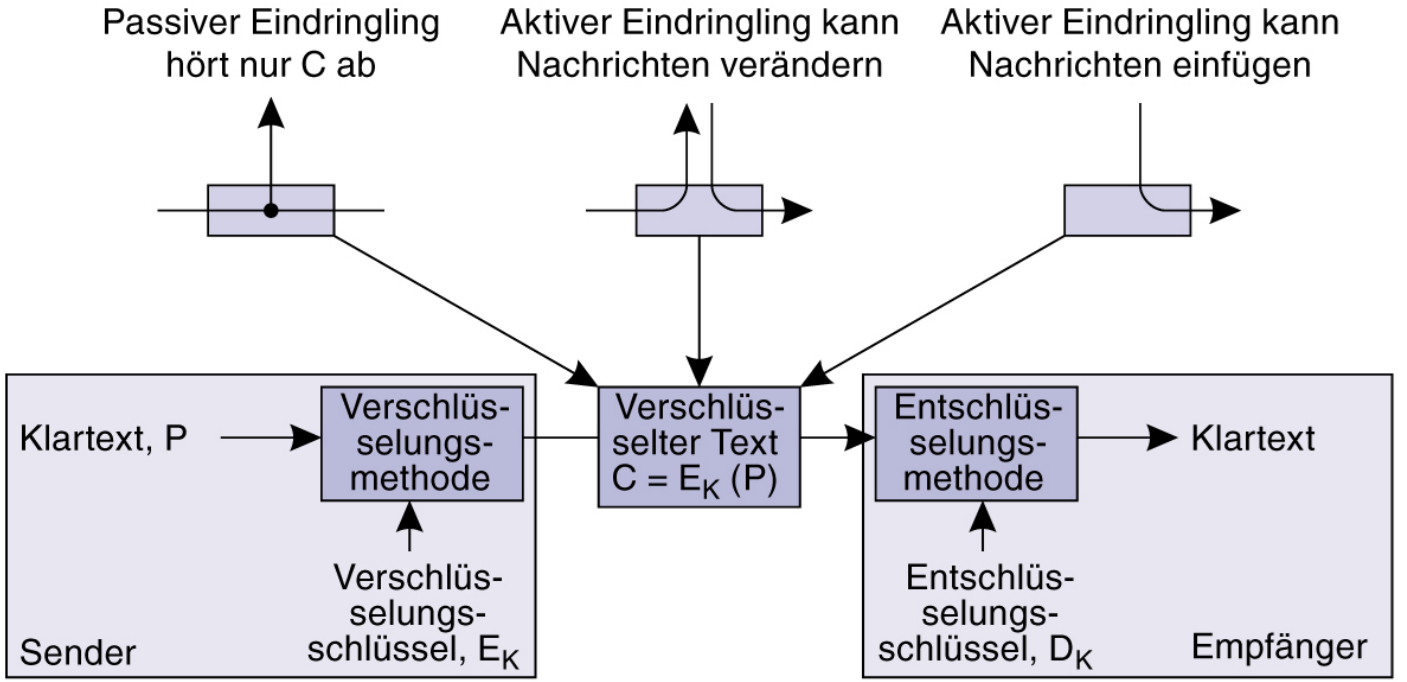
Kryptographie - Grundlagen

- Grundprinzip der Verschlüsselung
und Arten von Sicherheitsangriffen

- Klartext P

→ Verschlüsselter Text $C = E_K(P)$

→ Entschlüsselter Text $D_K(C)=D_K(E_K(P)) = P$



Kryptographie - Grundlagen

.Symmetrische Verschlüsselung

- Gleicher Schlüssel für die Ver- und Entschlüsselung (Sender und Empfänger haben gleichen Schlüssel)
- Begrifflichkeit: Secret-Key-Systeme, Shared-Key-Systeme
- Effizienz abhängig von Schlüssellänge – Bsp.: AES (Advanced Encryption Standard)

Schreibweise	Erklärung
$K_{A,B}$	Geheimer Schlüssel, den A und B gemeinsam nutzen
K_A^+	Öffentlicher Schlüssel von A
K_A^-	Privater Schlüssel von A

Kryptographie - Grundlagen

- **Asymmetrische Verschlüsselung**
  - Schlüsselpaar – öffentlicher Schlüssel und privater Schlüssel
  - Abhängig von Verwendung, welcher Schlüssel zu Ver- oder Entschlüsselung verwendet wird
    - Bsp.: Senden von Nachrichten: Verschlüsselung mit dem öffentlichen Schlüssel des Empfängers
    - Bsp.: Signaturen – Hashwert der Nachricht wird mit eigenem privaten Schlüssel verschlüsselt

Schreibweise	Erklärung
$K_{A,B}$	Geheimer Schlüssel, den A und B gemeinsam nutzen
K_A^+	Öffentlicher Schlüssel von A
K_A^-	Privater Schlüssel von A

Kryptographie - Grundlagen

- **Hash-Funktionen**
  - Hash-Funktion *H* erzeugt aus einer Nachricht *m* von beliebiger Länge eine Bit-Folge / Hash-Wert *h* mit fester Länge: $h = H(m)$
  - Eigenschaften
    - Einwegefunktionen → Bestimmung von *m* aus *h* durch Berechnung nicht möglich
    - Schwache Kollisionsresistenz → Keine Berechnung *m*‘ aus *m* möglich, so dass $H(m') = H(m)$
    - Starke Kollisionsresistenz → Aus Kenntnis von *H* ist es durch Berechnung nicht möglich zwei unterschiedliche Eingaben *m* und *m*‘ zu finden, so dass $H(m') = H(m)$

Kryptographie - Grundlagen

- **Hash-Funktionen**
  - Beispiel: md5
    - „Das ist ein Test.“ → e2f8af8ce3fa850a1591100a57f12222
    - „Das ist ein Dest.“ → 58d5dd4a294adef74cff290e4096da12
    - Achtung: Hash-Kollisions bekannt
    - Unsicher aufgrund Existenz einer „Rainbow Table“ (z.B. 1-7 Zeichen und Erfolgschange von 99,9% → Tabelle umfasst 52 GB)

Interludium: RSA-Verschlüsselung

- Benannt nach 3 Mathematikern: Rivest, Shamir, Adleman
- Vorausberechnung

1) Wähle zufällig und stochastisch unabhängig 2 Primzahlen

$$p \neq q$$

2) Berechne RSA-Modul:

$$N = p * q$$

3) Berechne Eulersche Funktion:
(Anzahl der zu N teilerfremden ganzen Zahlen kleiner N)

$$\varphi(N) = (p - 1)(q - 1)$$

4) Wähle teilerfremde Zahl e zu $\varphi(N)$ mit

$$1 < e < \varphi(N)$$

5) Berechne Entschlüsselungsexponent d mit

$$(e*d) \bmod \varphi(N) = 1$$

Interludium: RSA-Verschlüsselung

- Verschlüsselung mit (e,N)
  - Unverschlüsselte Nachricht m
  - Verschlüsselung durch: $c = m^e \ mod \ N$

- Entschlüsselung mit (d,N)
  - Verschlüsselte Nachricht c
  - Entschlüsselung durch: $m = c^d \ mod \ N$

Interludium: RSA-Verschlüsselung

- Beispiel:
  - Vorausberechnungen
    1) $p = 7, q = 11$
    2) $N = p * q \rightarrow N = 77$
    3) $\varphi(N) = (p - 1)(q - 1) \rightarrow \varphi(N) = 60$
    4) $1 < e < \varphi(N)$ … $e = 13$ (ausgewählt)
    5) $(e*d) \bmod \varphi(N) = 1 \rightarrow d = 37$,
       da $(13*37) \bmod 60 = 481 \bmod 60 = 1$
  - Verschlüsselung: $m=2 \rightarrow c = 2^{13} \bmod 77 = 30$
  - Entschlüsselung: $c = 30 \rightarrow m = 30^{37} \bmod 77 = 2$
  - Hinweis: $(x^a)^b \bmod N = (x^a \bmod N)^b \bmod N$
  - z.B. $30^3 \bmod 77 = ((30*30) \bmod 77) * 30 \bmod 77 = (53 * 30) \bmod 77 = 50$
    $30^4 \bmod 77 = (30^3 \bmod 77) * 30 \bmod 77 = (50 * 30) \bmod 77 = 37$
    ….