Verteilte Systeme - Zusammenfassung

Christopher Mutschler

July 31, 2008

Contents

1 Disclaimer

Diese Zusammenfassung enthält noch einige Rechtschreibfehler. Das ist mir bekannt aber gewissermassen egal, da es nun vorbei ist. Lesen auf eigene Gefahr *g*

2 VS02.pdf - Bestandsaufnahme Nr. 1

Problembereiche:

- lokal → entfernt
  Im Falle entfernt ausgelegeter Interaktionen sind mehr Fehlerarten möglich als nur im lokalen fall
- direkte → indirekte bindung
  Konfigurierung wird zu einem dynamischen Vorgang und erfordert Bindungsunterstützung zur Laufzeit
- sequentielle → nebenläufige Ausführung
  Wirkliche Nebenläufigkeit (d.h. Parallelität) erfordert Mechanismen zur Sequentialisierung der Aktivitäten

- synchrone → asynchrone Interaktion
  Verzögerung durch die Kommunikation erfordern Unterstützung für asynchrone Interaktionen und zur Fliessbandverarbeitung (pipelining)
- homogene → heterogene Umgebung
  Interaktion zwischen entfernten Systemen erfordern eine gemeinsame Datenrepräsentation
- einzelne Instanz → replizierte Gruppe
  Replikation kann Verfügbarkeit und / oder Zuverlässigkeit bereitstellen erfordert aber auch Massnahmen zur Konsistenzwahrung
- fester Platz → Wanderung
  Die Lage entfernter Schnittstellen (zu Funktionen, Objekten, Komponenten) kann sich zur Laufzeit ändern
- einheitlicher → zusammengeschlossener Namensraum
  Die Namensauflösung muss (ggf. bestehende) Verwaltungsgrenzen zwischen verschiedenen entfernten Systemen reflektieren
- gemeinsamer → zusammenhangloser Speicher
  Mechanismen des gemeinsamen Speichers sind nicht (oder nur sehr eingeschränkt) im grossen Massstab anwendbar und wo entfernte Operationen involviert sind.

3 VS03.pdf - Bestandsaufnahme Nr.2

Heterogenität:

- Netzwerke: Anschlusstyp, Medium, Technik, Topographie
- Prozessoren: Informationsdarstellung, Byte Sex
- BS: Ausführungsumgebung, API
- Sprache: Semantik, Pragmatik
- Personen: Standards

einheitliches Programmiermodell zur Entwicklung verteilter Software:

- Prozedurfernaufruf (RPC)
- Objektfernaufruf (RMI)
- entfernte Ereignisbenachrichtigung oder SQL-Zugriffe
- verteilte Transaktionsverarbeitung

"Mobiler Code": Java-Applets, NOT(.exe-Dateien)
Nebenläufige Zugriffe auf verschiedenen Ebenen:

1. mehrere Prozesse benutzen einen Server zum gleichen Zeitpunkt
2. der Server ist Multithreaded.
3. Betriebsmittel liegen clientseitig als Replikate vor (Caching)

Was ist ein Fehler?

- FAULT unerwünschter Zustand, could lead to error
- ERROR Systemzustand entgegen der Spezifikation
- FAILURE Diensterbringung nicht mehr möglich

Klassifikation von Fehlern:

Gutmütige Fehler:

- CRASH STOP: Knoten fällt komplett aus
  - FAIL STOP: Jeder Knoten bekommt das mit
  - FAIL SILENT: eben nicht
- CRASH RECOVERY: Ein Knoten kann endlich oft ausfallen und wieder anlaufen

Bössartige Fehler:

- naja, schicken eben Informationen rum, die falsch sind
- oder: bewusst falsche Daten (Hack etc.)
- Fehler erkennen, maskieren oder tolerieren können! (Redundanz)

Sicherheit:

- Schutz vor Offenlegung → Vertraulichkeit
- Schutz vor Veränderung → Integrität
- Schutz vor Störungen → Verfügbarkeit

Offenheit von Systemen:

- offene Systeme:
  - Veröffentlichung der Schnittstellen
- offene verteilte Systeme:
  - einheitlicher Kommunikationsmechanismus
  - veröffentlichte Schnittstellen
  - Zugriff auf gemeinsame Betriebsmittel
  - Heterogene oder homogene Hard- und Software

"Ein System, das als skalierbar bezcihnet wird, bleibt auch dann effektiv, wenn die Anzahld er Ressourcen und die Anzahl der Benutzer wesentlich steigt.

Zugriffstransparenz:

ermöglicht den Zugriff auf lokale und globale (entfernte) Betriebsmittel unter Verwendung identischer Operationen. Das betreffende API macht keinen Unterschied zwischen lokalen und entfernten Operationen.

Ortstransparenz:

erlaubt den Zugriff auf Betriebsmittel, ohne ihre Position kennen zu müssen. Beispielsweise erfolgt der Zugriff nicht direkt über IP sondern über DNS

Nebenläufigkeitstransparenz: gleichzeitiges, konfliktfreies Arbeiten

Replikationstransparenz mehrere Betriebsmittelinstanzen, zur Leistungsverbesserung

Fehlertransparenz kontinuierlichen Rechnereinsatz (non-stop-computing)

Migrationstransparenz: Verschieben von Betriebsmitteln / Prozessen ohne Beeinträchtigung

Leistungstransparenz: Lastabhñgige Neukonfigurerung zur Leistungssteigerung

Skalierungstransparenz: Vergrösserung / Verkleinerung ohne Beeinträchtigung eingesetzter Algorithmen

Unschärfeprinzip:

> "Es ist die Eigenart verteilter Systeme, dass es auf Dauer keine zwei Prozesse gibt, die zur gleichen Zeit die gleiche, zutreffende Sicher des Systems haben. Ein Prozess innerhalb des Systems verfügt entweder über vollstängige, aktuelle oder über vollständige, überholte Zustandinformation.

1. Keine gemeinsame physikalische Uhr
2. Keinen gemeinsamen Speicher
3. Keine akkurate Ausfallerkennung

4 VS04.pdf - Kommunikationssystem

IPC Semantiken

1. no-wait-send: der Sendeprozess wartet, bis die Nachricht im
   - Transportsystem zum Absenden bereitgestellt worden ist
   - (Pufferung oder Signalisierung bei Freiheit des Puffers)
2. synchronization send:
   - der Sendeprozess wartet, bis die NAchricht vom Empfangsprozess angenommen worden ist
   - (Rendezvous zwischen Sender/Empfänger)
3. remote-invocation send
   - der Sendeprozess wartet, bis die Nachricht vom Enpfangsprozess verarbeitet und beantwortet worden ist
   - (Fernaufruf einer vom Empfangsprozess auszuführenden Funktion)

IPC Varianten:

- non-blocking send → no wait send
- blocking send → der Sendeprozess wartet, bis die Nachricht den Rechner verlassen hat, d.h. bis die ausgegeben und ins Netz eingespeist wurde.
- reliable-blocking send → der Sendeprozess wartet, bis die Nachricht beim Empfangsrechner eingetroffen ist, bzw. von dem Empfangsprozess-BS angenommen wurde.
- explicit-blocking send → synchronization-send
- request/reply → remote-invocation send

IPC-Protokolle für Fernaufrufe:

- request (R):

  > kann genutzt werden, wenn die entfernte Prozedur keinen Rückgabewert liefert und der Sendeprozess keine Bestätigung für die erfolgte Ausführung benötigt. (1)

- reqest-reply

  > ist das geläufigste Verfahren, da die Antwortnachricht implizit die Anforderungsnachricht bestätigt und dadurch explizite Bestätigungen entfallen (2)

- request-reply-acknowledge reply (RRA)

  > gestattet es, die zum Zwecke der Fehlermaskierung (beim Server) gespeicherten Antwortnachrichten zu verwerfen, wenn (vom Client) keine weitere Anforderungsnachricht gesendet wird. (3 Nachrichten)

Kommunikationendpunkte:

- PROZEDUR → das die Nachricht verarbeitende passive Objekt (method)
- PROZESSINKARNATION → das verarbeitende aktive Objekt (thread)
- TOR → das Nachricht weiterleitende Objekt (port)
- BRIEFKASTEN → das die Nachricht zwischenspeichernde Objekt (mailbox)

Kommunikationsverlauf:

1. direkt: der Bezeichner identifiziert einen Prozess → direkte Zustellung
2. indirekt: an ein Tor oder einen Briefkasten
3. verbindungsorientiert: der Bezeichner identifiert ein Tor; Verbindung zwischen zwei Toren

Möglichkeiten / Varianten der Interprozesskommunikation:

- asynchron / synchron
- ungepuffert / gepuffert
- nicht-blockierend / blockieren
- unzuverlässig / zuverlässig

- no-wait send
- blocking send
- reliable-blocking send
- synchronization send
- remote-invocation send
- direkt, indirekt oder verbindungsorientiert!

5 VS05.pdf - Fernaufrufe

> "Eine Kommunikation, bei der die bei einem anderen Prozess beantragte Aktivität im Vordergrund steht, wird aktionsorientiert genannt."

→ Aktionsorientierte Kommunikation meint "Prozedurfernaufruf"

Asynchrone Varianten: "Versprechen" (Promise)

- der RPC kehrt sofort zurück
- ein promise-Objekt dient zur Aufnahme eines resultats
- der Objektzustand kann mittels ready() abgefragt werden
- linguistische Unterstützung ist angebracht
- mit no-wait-send wird die promise-Implementierung ideal unterstützt

Client-Stub: Prozedurstumpf auf Seite des Auftraggebers

- abstrahiert von der Örtlichkeit der fernen, aufgerufenen Prozedur
- setzt den Prozeduraufruf um in einen Nachrichtenaustausch
- verpackt aktuelle Parameter und entpackt Rückgabewerte

Server-Stub: Prozedurstumpf auf Seite des Auftragnehmers

- abstrahiert von der Örtlichkeit der fernen, aufrufenden Routine
- setzt die Prozedurrückkehr um in einen Nachrichtenaustausch
- entpackt aktuelle Parameter und verpackt Rückgabewerte

Verklemmungsfreie Fernrückrufe auf zwei Arten realisierbar

1. Zustandmaschine (state machine)
   - Synchronität durch synchronization send oder n-wait send lockern
   - nach dem Senden, mit receive auch Antwort oder Rückruf warten
   - der Faden muss isch zwischen verschiedenen Arbeitsphasen multiplexen
2. Rückrufanbieter (back-call server)
   - remote-invocation send - Modell als Fernaufrufgrundlage beibehalten
   - zur Rückrufverarbeitung einen eigenen speziellen Faden vorsehen
   - den Rückruf faktisch als ganz "normale" Dienstleistung verstehen

⇒ Die Lösung 2 ist orthogonale Lösung, strukturfördernd und weniger fehleranfällig!

Semantikaspekte im Vergleich konventioneller Funktionsaufrufe und verteilter:

- Parameterarten: Eingabe- / Ausgabe-/ Ein-Ausgabeparameter
- Parameterübergabe: call-by-reference vs call-by-value/result
- Gültigskeisbereiche schränken sich ggf. massiv ein: Variablen des fernen umfassenden scopes sind nicht zwingend gültig
- Speicheradressen zu versenden ist (nahezu) sinnlos

→ Eingabeparameter: NUR Bestandteil der Aufforderungsnachricht
→ Ausgabeparameter: NUR Bestandteil der Antwortnachricht
→ Ein-/Ausgabeparameter: Bestandteil beider Nachrichten!

Problem in C/C++: Zeiger, Referenz, Feld <–> Welche Parameterart? → Die Parameterart muss (IDL) spezifiziert sein!
call-by-value/result: wie beim konventionellen Fall!

call-by-reference:

- Eingabeparameter → call-by-value → dereferenzieren
- Ausgabeparameter → call-by-result → dereferenzieren
- Ein-/Ausgabeparameter → call-by-value/result → dereferenzieren
- unspezifiziert → call-by-value SPEICHERADRESSE

call-by-name (eine Funktion wird mitgeliefert, die den aktuellen Parameter berechnet)

Ferne Logische Adresse:

eindimesionaler Adressraum (paging):

- $0815 muss im (logischen) Adressraum des Anbieters frei sein zweidimensional Adressraum (segmentation)
- $0815 ist ein Versatz (offset) innerhalb einer Segments
- mit Empfang der Adresse wird die Adresse 0815 definiert, d.h. lokal eindeutig
- der Adressraum des Anbieters verändert sich je nach Fernaufruf

LÖSUNG: SMART POINTER!

Zweiteiliger (getypter) "geschickter Zeiger"

- Kontextbezeichner systemweit eindeutig (zb Prozessinkarnation)
  - durch Rückruf wird das referenzierte Objekt per Ferne herangeholt
  - ein Platzhalter zur Aufnahme der lokalen Kopf wird dynamisch angelegt
  - die Adresse des Platzhalters wird als Zeigerkomponente übernommen
- Zeiger Adresse auf (a) das ferne Objekt, (b) den lokalen Platzhalten

  aber: ohne linguistische Unterstützung ist die Technik wenig praktikabel:

C++

- ferne Adressen z.B. als Instanzen von Klassenschablonen realisieren
- überlagerte Operatoren setzen bei Bedarf Rückrufe ab und regeln den Zugriff

ggf. sind lokale Kopien nicht zwingend und alle Zugriffe können Rückrufe sein!

Nachrichten zusammenstellen/auseinandernehmen: (marshalling/unmarshalling)

1. Die Datenposten in einen Nachrichtenpuffer gepackt anordnen (Serialisieren, Repräsentation!)
2. Ja nach Herangehensweise sind referenzen auszulösen (falls call-by-value)
3. die Nachricht als Paket per IPC zum Empfänger schicken

Unmarshalling: inverse Funktion dazu.

Objektorientierung erschwert das ganze auch noch: Erst zur Laufzeit ist bekannt ob ein Objekt eine spezialisierten Unterklasse angehört, daher scheidet eine automatische Generierung des Stumpfes zur Übersetzungszeit aus. $\longrightarrow$ Vollständige Analyse des zu verteilenden Quellprogrammes ist notwendig!

Konvertierungsoptionen:

- beidseitig (external data representation - XDR):
  - beideitig in eine kanonische Basis bringen -> Nicht notwendig bei gleichartigen Rechners
- sendeseitig (sender makes it right)
  - zu sendende Daten in die empfangsseitige Darstellung ggf. umkodieren
  - Problem: Mehrteilnehmerkommunikation, Weiterleiten von Nachrichten
- empfangseitig: (receiver makes it right)
  - empfangene Daten in die lokale ggs umkodieren (->endian tag)

EXTERNAL DATA REPRESENTATION (XDR)

- Daten werden als "Vielfaches von vier Bytes" repräsentiert
- beim Lesen/Schreiben eines Stroms kommt Byte m immer vor m+1
- Füllbytes (0-3) ergänzen den Datenstrom immer zum Vielfachen von 4
- Die Reihenfolge (der Byte-Sex) is big endian

```
big  <-->  big optimal
big  <-->  little
little  <-->  big
little  <-->  little suboptimal
```

→ VERSCHNITT!

Sprachintegration von Fernaufrufen:

1. Von der Programmiersprache integriertes Konzept:
   - internes Wissen über Datentypen und Lautzeitmodell
   - der Compiler erzeugt auch Stümpfe
   - umfassende Analysen, Optimierungen und Automatismen möglich
2. Von der Programmiersprache unabhängig (CORBA)
   - das Schnittstellenverhalten ist in der IDL zu definieren
   - fehlendes internes Wissen schränkt Analyses und Optmierungen ein

Zustellungsgarantien: → request-reply Protokolle

1. Wiederholung der Aufforderungsnachricht bis die Antwort eintrifft oder ein Ausfall angenommen wird → request retry
2. Filterund der Aufforderungsduplikate, wenn die Aufforderung empfange wurde und noch in Arbeit ist → duplicate supression
3. Wiederholung der Antwortnachricht, bis die nächste Aufforderung oder eine Bestätigung eintrifft → reply retry

	OPTION		SEMANTIK	
		re-execute		
request retry	dupl.supr.	reply-retry		
NEIN	-	-	kann sein	maybe
JA	NEIN	re-execute	min. einmal	at-least-once
JA	JA	reply-retry	max. einmal	at-most-once

weitere Aufrufsemantiken: last-of-many: akzeptiert nur die Antwort zum jüngst zurückliegenden Aufruf:

- jeder Aufruf, auch der wiederholte, erhält eine eindeutige Kennung
- jede Antwort trägt die Kennung des passenden Aufrufs
- veraltete Kennungen werden verworfen

exactly-once: genau einmal, entsprechend der lokalen Aufrufe

- erfordert Transaktionskonzepte nach dem "Alles oder nichts prinzip"
- → wünschenswerte, ideale Semantik, die in "Reinform" jedoch unerreichbar ist.

Verwaister Aufruf (Orphan)

Ein Klient, der den Aufruf (z.B. wegen eines ggf. zu kurzen Timeouts) abgebrochen hat und nicht mehr am Ergebnis interessiert ist oder der mittlerweile überhaupt nicht mehr zur Verfügung steht.

Massnahmen:

- zusätzliche Zeitüberwachung des Klienten durch den Dienstanbieter
- pro Klient inen Ausfallzähler beim Dienstanbieter verwalten (alte abbrechen)
- zusätzliche direkte Statusanfragen an den Klienten senden (schnelle Waisenerkennung, sofern diagnostizierbar)
- → Grosse Probleme bei FERNAUFRUFKETTEN!

Waisenbehandlung:

1. extermination (Vertilgung, Ausrottung, Wegschaffung)

   nach Wiederanlauf wird beim Klienten auf ausstehende aufrufe geprüft -> Abbruchanforderungen werden an den Anbieter geschickt; rekursives Vorgehen. Klientenstümpfe müssen Buch führen -> Log anlegen

2. expiration (Verscheiden, Ablauf)

   der ANbieter gibt dem Aufruf ein Zeitquantum Bei Ablauf erbittet der Anbieter ein weiteres Zeitquantum beim Clienten -> Abbruch im Fehlerfall -> Nach wiederanlauf is der erste Fernaufruf um ein Zeitquantum zu verzögern (???)

3. reincarnation (Wiederverkörperung, Wiedergeburt)

   Scheiter die Ausrottung wird eine neue Epoche eröffnet -> Mitteilung an alle Maschinen -> Alle Anbieter terminieren -> Jede Anforderung enthält eine Epochenkennung zum erkennen verwaister aufrufe

4. gentle reincarnation (sanft, milde)

   zum Epochenbeginn versucht der Anbieter seinen Klienten zu lokalisieren

6 VS06.pdf - Namensräume

Adresse: ermöglicht den effizienten Zugriff auf Objekte bzw. Betriebsmittel

Attribut: is allgemein der Wert einer einem Objekt zugeordneten Eigenschaft; ermöglicht die interne Struktur eines Unternehmens nicht offenzulegen relevantes Schlüsselattribut: Adresse!

Bindung: bezeichnet die Zuordnung zwischen einem Objekt und einem Namen

Name <–> Adresse: Eine Adresse kann auch als zusätzlicher Name betrachtet werden, der nachgeschlagen werden muss, aber auch einen solchen "Namen" enthalten kann. (IP-Adresse -> Netzwerkadresse, E-Mail-Adresse -> E-Mail-Server....)

Universal Resource Identifier (URI): stellt die Adresse einer WEB-Resource dar wird die Ressource verschoben / gelöscht, entsteht eine hängende Referenz

Uniform Resource Name (URN): löst das Problem hängender Referenzen entspricht einem migrationstransparentem dauerhaften Namen einer Web-Resource wird bei einem URN-Nachschlagedienst hinterlegt z.B. urn:Namensraum:Name -> urn:ISBN:0-13-18369-3

Universal Resource Characteristic (URC): ausschlieSSlich auf Attributen einer Resource basierend: author/title/keywords stellt eine URN-Untermenge dar, und ist eine Verfeinerung der Namensgebung ťwird mit URL (ggf. URN) bei einem URC Nachschlagedienst hinterlegt geliefert wird eine URL

Unterscheidung: hierarchische und flache Namensräume!

- flacher Namensraum:
  - keine interne Struktur (äuSSere Struktur schafft ggf. Lesbarkeit)
  - die Ausdehnung eines flachen Namensraums ist meist endlich
- hierarchischer Namensraum:
  - es existiert eine interne Struktur (z.B. Pfadnamen)
  - die Ausdehnung eines hierarchischen Pfadnamens ist pot. unendlich
  - → durch Einbindung von Kontexten

heterogener Namensraum: es ist möglich "." als Stammverzeichnis eines anderen Namensraums zu benennen.

Kombinierter Namensraum:

- der Namensraum kann ganz oder teilweise in einen anderen Namensraum eingebettet sein. (mount)
- das ALIAS Konzept hilft, komplexe Namensstrukturen zu vereinfachen

Namensalias:

- symbolischer Link: Domänennamen durch andere Domänennamen ersetzen
- ein Mittel zur Sicherstellung der Transparenz

Beispiel: www4.informatik.uni-erlangen.de -> faui2y.informatik.uni-erlangen.de Aliase sind von Namensdienst zu erkennen und entsprechend aufzulösen

Namensdomäne:

- Domäne als Begriff der administrativen Autorität
- left Verantwortlichkeit fest
- die Subdomäne informatik.uni-erlangen.de is relativ eigenständig (autonome Verwaltung)

Namensauflösung:

- iterativ: alle Anbieter kontaktieren, bis der Name aufgelöst werden kann
- Mulitcast: an eine Gruppe von Anbietern senden (Problem bei mehrmaliger Möglichkeit des Auflösens)

  Unterscheidung in Klientgesteuert und Anbietergesteuert (rekursiv, nicht-rekursiv)

  Caching: Zwischenlagerung von Namensauflösuingen verkürzen die Antwortzeit

Nachschlagedienst:

- Namensdienst:speichert eine Auflistung von Bindungen zwischen Namen und Attributen aus einem oder mehreren Namenskontexten.
- Verzeichnisdienst: attributbasierter Namensdienst zum suchen anhand von Attributen
- Erkennungsdienst: Verzeichnisdienst zur Registrierung von bereitgestellten Betriebsmitteln
- Verzeichnisdienst: sucht nach Einträgen von Namen, die bestimmten Attributen entsprechen (Attribute sind für die Bezeichnung von Objekten leistungsfähiger als Namen)

7 VS07.pdf - Grundlagen

Aus Sicht der Fehlertoleranz:

- Keine gemeinsame Uhr
- Kein gemeinsamer Speicher
- Keine akkurate Ausfallerkennung

Eigenschaften eines "perfect link":

1. Zuverlässige Übermittlung
2. Keine Verdopplung
3. Keine Erfindung

Eigenschaften eines "fair-loss-link":

1. faire Verluste (irgendwann kommts mal an)
2. Endliche Verdopplung
3. keine Erfindung

Multicast-Kommunikation:

- Kausale Ordnung: a tritt vor b ein wenn b durch a beeinflusst werden konnte
- Konsistente totale Ordnung: Es gibt eine systemweite totale Ordnung der Nachrichten
- Kausale totale Ordnung: Konsistene Ordnung + Kausale Ordnung
- Physikalische Ordnung: Ordnung nach Realzeit

Synchron vs. Asynchron

- Entfernte Methodenaufrufe:
  - synchron: Aufrufer wartet
  - asynchron: Aufrufer blockiert sich nicht
- Ausführung von verteilter Aktivitäten:
  - synchron: Synonym zu "gleichzeitig"
  - asynchron: Keine Synchronisierung
- Koordinierung
  - Synchronisierung für "nicht-gleichzeitig" gebraucht

Vorteile synchroner Systeme:

- Ablauf in Runden
- Ausfallerkennung
- Zeitbasierte Koordinatino
- Synchronisation physikalischer Uhren
- Performance-Analyse möglich

Nachteile:

- In vielen Fällen (internetbasiert) realitätsfern

Ausfallerkennung

- Beliebte Abstraktion fü Algorithmen im verteilten System
  - Orakel liefert Aussagen über den Zustand von Rechnern
  - Orakel erfüllt formal fassbare Eigenschaften
- Ziele / Vorteile
  - Vereinfachung von Algorithmen durch Modularisierung
  - Algorithmen können ohne der Realisierung der Ausfallerkennung implementiert und verifiziert werden

Analyse von Algorithmen

- Sicherheit: Der Algorithmus liefert keine falschen Ergebnisse
- Lebendigkeit: Der Algorithmus verklemmt sich nicht

Bewertung nach...

- ...Zeitauwand
- ...Nachrichtenaufwand
- ...Verifizierbarkeit
- ...Implementierungsaufwand

8 VS08.pdf - Uhrensynchronisation

- Logische Uhren nach Lamport: kausale Abhängigkeit: a $\longrightarrow$ b
- Vektoruhren (auch logisch): jeder Prozess hat einen Zeitvektor in der die letzten Zeitstempel der jeweils anderen Prozesse stehen...

8.1 NTP - Network Time Protocol

- Entwickelt seit 1982 von David Mills
- 151 aktive Stratum 1
- 208 aktive Stratum 2
- Genauigkeit: 0.01ms in WANs, $<$0.1ms in LANs
- fehlertolerant

Grundlegender Überblick

- Stratum 1 über Funk oder Leitung an Zeitstandards
- hierarchisches Netz
- Zuverlässigkeit durch Redundanz
- verschiedene Betriebsarten (Master/Slave, symmetrische Sync., Multicast, Authentifizierung...)

Wichtige Eckdaten

- Peer-Filter wählen pro Ref.-Server den besten Wert aus den acht letzten Offset-Messwerten
- Die Algorithmen versuchen richtige Uhren zu erkennen und falsche auszufiltern
- Kombinationsalgorithmus: gewichteter Mittwert der Offsetwerte
- Loop-Filter und VFO (Variable Frequency Oscillator) bild die geregelte lokale Uhr. Jitter und Drift werden bei der Regelung minimiert.

9 VS09.pdf - Wahlalgorithmen

Problemstellung:

- Es gibt Situationen, die einen Anführer erfordern
  - Koordinator von verteilten Aktionen
  - Erzeugung eines eindeutigen Token
  - ...
- Automatische Bestimmung muss sein!
  - Initial beim Start
  - Bei Neukonfigurierung nach Fehlern

Formale Anforderungen an einen Wahlalgorithmus:

- Eindeutigkeit (Es wird ein einziger Knoten bestimmt)
- Terminierung (In endlicher Zeit fühlt sich ein Knoten als Anführer)

9.1 Chang-Roberts-Algorithmus

Vorraussetzung:

- Jeder Knoten besitzt eine eindeutige ID; totale Ordnung

Ziel:

- Finde Knoten mit maximaler ID

Ablauf: Start: Jeder sendet seine eigene ID weiter.
Beim Empfang einer ID:

1. Falls empfangene ID grösser als eigene: weitersenden; Knoten nicht mehr wählbar
2. Falls IDs gleich: als Anführer gewählt
3. Falls empfangene ID kleiner: Nachricht ignorieren

Komplexität:

- Zeit O(n)
- Nachrichten O(n^s)

Verbesserung: Entweder nach links oder rechts (Zufall) die Nachricht senden. Bei ausgeglichener Wahrscheinlichkeit ist O(n log n) möglich.

9.2 Peterson-Algorithmus

Ziel:

- Verringerung der Nachrichtenzahl im worst-case auf O(n log n)

Zustandsvariablen:

- Zustand: aktiv, weiterleiten, gewählt; inital: aktiv.
- ID: eine Knoten-ID; initial: eigene ID

Ablauf in Phasen:

1. Jeder aktive Prozess i sendet den Wert seiner Variablen ID an die beiden nächsten aktiven Prozesse im Ring weiter
2. Jeder aktive Prozess vergeleicht seine ID, mit der von den beiden aktiven Vorgängern empfangenen ID:
   - hat der direkte Vorgänger die grösste ID, so bleibt der Prozess i aktiv und übernimmt die ID
   - ansonsten wechselt der Prozess in den Zustand "weiterleiten"
3. Wenn die vom direkten aktiven Vorgänger empfangene ID gleich der eigenen ID-Variablen ist, so ist er der einzige verbleibende aktive Knoten und somit der Anführer

Komplexität

- Nachrichten O(n log n)
- Zeit O(n log n); log n Phasen mal n Nachrichten (genauer: O(n))

9.3 Wahl auf Bäumen

Grundidee:

- Traversieren des kompletten Baums, um die grösste ID zu finden
- sequentiell
- parallel (sehr gut für verteilte Systeme ;))

Aufwand bei sequentiellem Suchen: 2(N-1) Kommunikationsschritte!

Wellenverfahren:

- Explosionswelle: Ein Starter sendet eine Explosionsnachricht in alle Richtungen; innere Knoten tun das ebenfalls falls sie das erste mal eine erhalten;
- Echowelle: Ein Blattknoten, welcher eine Explosionsnachricht erhählt sendet eine Echowelle zurück (mit seiner ID)
- Informationswelle

Innerer Knoten: Ein Innerer Knoten, der auf k-1 von k Kanten Echonachrichten empfangen hat, sendet eine Echonachricht mit der höchsten ID an die k-te Kante weiter.

Sonderfall: Erhält ein Knoten nach Versenden seiner eigenen Echonachricht eine weiter Echonachricht, so ist das Maximum beider IDs die höchste ID-Nummer im Netz.

- Muss lediglich ein beliebiger Anführer bestimmt werden, so kann man jetzt einfach einen der beiden Knoten auswählen
- Ansonsten kann man in einer Informationswelle alle Knoten über die ermittelte höchste ID informieren

9.4 Wahl auf beliebigen Topologien

sequentielle Traversierung: Jeder Baumknoten, beim ersten Empfang des Tokens:

- Falls zuvor bereits ein Token mit grösserer ID empfangen wurde, wird das Token verworfen
- Ansonsten sendet der Knoten das Token zu einem Nachbarn, der das Token bisher noch nicht besessen hat
- Wenn es keinen solchen Nachbarn mehr giobt, wird das Token über die Kante zurückgesendet, über die es zuerst empfangen wurde.

Erster Empfang des Tokens: Mehrfacher Empfang des Tokens über verschiedene Pfade (Schleifen) muss verhindert oder erkannt werden!

Weiter Verfahren:

- Pfadverfahren
  - Jeder Knoten kennt die Identität seiner Nachbarn, und im Token wird der komplette bisher durchlaufenen Pfad gespeichert
  - Jeder Knoten kann selbst entscheiden, welche Nachbarn das Token noch nicht besessen haben
- Kantenfärbungsverfahren
  - Die Knoten müssen die Identitäten der Nachbarn nicht kennen, im Token wird keine Pfadinformation gespeichert
  - Das Token muss über alle Kanten gesendet werden, um festzustellen, ob der Nachbar das Token bereits besessen hat. Bereits verwendete Kanten werden markiert ("gefärbt"), um jede Kante genau einmal zu verwenden.

Jetzt: Parallel Traversierung!

9.4.1 Echo-Election-Algorithmus

Ablauf:

1. Ein freier Knoten i wird spontan zum Initiator und sendet ein init(i) an alle Nachbarn
2. Ein freier Knoten bekommt eine init(i)-Nachricht
   - Grad > 1: markiere dich selbst als "i" und schicke init(i) an alle deine Nachbarn
   - Grad $= 1$: Echonachricht zurückschicken
3. init(j) trifft auf einen mit i markierten Knoten (Grad > 1):
   - j = i: zwei Explorer identischer Markierung begegnen sich; Kante löschen!
   - j < i: der eintreffende Explorer ist kleiner als der zuletzt versandte: Nachricht verwerfen!
   - j > i: der Knoten wird erobert; Falls ein Initiator erobert wird, entfällt damit dessen virtuelle Baumkante.
4. Es trifft ein Echo ein:
   - i = j: Falls die Nicht-Baumkante als erledigt festgestellt wird, wird ein Echo verschickt
   - i > j: Nachricht wird ignoriert.
5. Zurücksenden des Echos: Falls es sich um einen Initiator handelt (d.h. Baumkante ist virtuell), so ist er der Gewinner.

Echo/Election-Adoptionsverfahren:

1. Ein freier Knoten wird Initiator; **ausgehende Kanten werden mit i markiert**
2. Ein freier Knoten wird von einem Explorer ε(i) erreicht:
   - Grad > 1: wie bisher; **ausgehende Kanten werden mit i markiert**
   - Grad $= 1$: wie bisher
3. Ein Explorer ε(j) trifft auf Knoten mit Markierung i (Grad > 1); die Eingangskante sei mit k markiert:
   - k = j: zwei Explorer begegnen sich (mit gleicher Markierung)
   - k > j: der eintreffende Explorer ist kleiner als der in Gegenrichtung versandte: keine Änderung
   - k < j: (damit natürlich auch j > k): **Adoption:** Explorer wird über die bisherige Baumkante versendet. Falls der Initiator erobert wird, wird ggf. ein Echo zurückgesendet.
   - i $\geq$ j > k: Falls i = j: Baumkante erledigt; Dieser Fall tritt nur auf, wenn es mind. 3 Initiatoren gibt.
4. Es trifft ein Echo ein: wie gehabt.
5. Zurücksenden des Echos: wie gehabt.

10 VS10.pdf - Gegenseitiger Ausschluss

Prinzipielle Möglichkeiten:

- Zentraler Koordinator
- Vollständig verteilte Lösung (Lamport, Ricart und Agrawale)
- Quorumbasiert: Dezentral, Jeder fragt ausreichend viele (Maekawa, Sanders)
- Tokenbasiert: Wer es hat, der darf; Token wandert von allein (Perpetuum-Mobile, Suzuki und Kasami, Raymond, Singhal)

Formale Anforderungen an den Gegenseitigen Ausschluss:

1. Gegenseitiger Auschluss
2. Lebendigkeit bzw. Verklemmungsfreiheit (e.i. no deadlocks)
3. Strenge Anforderung: Aushungerungsfreiheit

Weitere Gesichtspunkte:

- Fairness
- Fehlertoleranz
- Erzeugte Netzlast
- Erzielte Performanz

Definition: Koordinierungsverzögerung

> Diejenige Zeit, die vergeht nachdem ein kritischer Abschnitt verlassen wurde bis zu dem Zeitpunkt wo er wieder betreten werden darf.

10.1 Zentraler Koordinator: Lamport:

Nachrichtentypen:

`REQUEST - REPLY - RELEASE`

- Eintrittswunsch des Knotens P_i: sende REQUEST (t, i) an alle ausser dich selbst
- Knoten P_i verlässt den kritischen Abschnitt: sende RELEASE(t, i) an alle

Bearbeitung von Request-Nachrichten:

- Jeder Prozess sammelt diese nach geordnetem Zeitstempel lokal
- Wenn P_j einen REQUEST von P_i erhält wird ein ACK gesendet.

Betreten / Verlassen des kritischen Abschnitts:

- Der Prozess darf den kritischen Abschnitt betreten, wenn sein REQUEST am Kopf der Warteschlange steht, oder wenn er von allen REQUESTS den kleinsten Zeitstempel hat
- Verlassen: sein REQUEST aus der Warteschlange entfernen und ein RELEASE an alle Knoten senden; Wenn P_j ein Release von P_i bekommt, entfernt er dessen REQUEST aus der lokalen Warteschlange

Komplexität: T=Koordinierungsverzögerung

- Nachrichten: 3(N-1) pro k.A.
- Antwortzeit: 2T + E
- **Keine Fehlertoleranz: beim Ausfall eines Knotens deadlock!**

10.2 Zentraler Koordinator: Ricart und Agrawala

Trick: Kombination von ACK und RELEASE!

REQUEST-Bearbeitung:

- Wenn j ein REQUEST bekommt, dann sendet er REPLY wenns ihm egal ist oder wenn er auch will aber sein Zeitstempel später liegt
- In allen anderen Fällen das Senden aufschieben

Betreten / Verlassen kritischer Abschnitte:

- Betreten: Wenn er von allen ein REPLY bekommen hat
- Verlassen: sende REPLY zu allen aufgeschobenen REQUESTS

Komplexität

- Antwortzeit: 2T + E (E=mittlere Ausführungszeit)
- Nachrichten: 2(N-1)

Optimierung von Carvalho und Roucairol:

- Implizietes Weiterverwenden von REPLY-Nachrichten (Erlaubnis)
- Wenn REPLY von i kam und kein REQUEST bisher will er auch nicht in den k.A.
- Anzahl der Nachrichten Variabel: 0...2(N-1)