Der Kerberos-Sicherheitsdienst

- Protokoll zur Schlüsselvergabe, Authentifizierung und Einrichtung sicherer Kommunikationskanäle
- Am MIT entwickelt im Rahmen eines erstes grossen Client-Server-Campusnetzes
  - war dort ab 1986 im Einsatz
- Basiert auf Needham-Schroeder-Protokoll mit symmetrischen Schlüsseln (z.B. DES)
- Public domain; es gibt auch kommerzielle Varianten
- In heutigen "offenen" verteilten Systemen gibt es noch andere Systemdienste zur Erhöhung der Sicherheit
  - z.B. ssh, VPN etc.

R.M. Needham, M.D. Schroeder: *Using Encryption for Authentication in Large Networks of Computers.* CACM 21(12), pp. 993-999, 1978

J.I. Schiller: *Sicherheit im Daten-Nahverkehr.* Spektrum der Wissenschaften 1/1995, pp. 50-57, Januar 1995

B. Clifford Neuman and Theodore Ts'o: *Kerberos: An Authentication Service for Computer Networks.* IEEE Communications Magazine, Volume 32, Number 9, pp. 33-38, September 1994

RFC 1510: *The Kerberos Network Authentication Service (V5),* http://www.rfc-archive.org/getrfc.php?rfc=1510

Kerberos-"Philosophie"

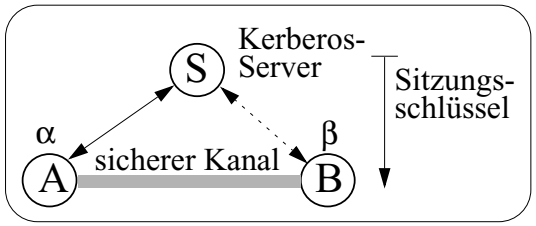
- Offenes Campusnetz → Nachrichten prinzipiell unsicher
- Kommunikation daher i.Allg. verschlüsselt und nur mit authentifizierten Partnern
  - Kenntnis des Sitzungsschlüssels als Authentitätsbeweis
- Passwörter niemals im Klartext übertragen
  - auch keine Passwortspeicherung

"Kerberos-Server"

- Benutzer, Clients und Server sind bei zentraler Instanz (Key Distribution Center: "KDC") akkreditiert
  - vereinbaren mit dem KDC auch ihren Geheimschlüssel ("master key")
  - ohne Akkreditierung keine Server-Berechtigungsscheine ("Tickets")
  - ohne Tickets kein Service
  - Ticket nur in Verbindung mit Authentitätsnachweis gültig
- Gültigkeit von Tickets / Sitzungsschlüsseln zeitlich befristet
- Mehrere Sicherheitsstufen möglich
  (1) Authentifizierung nur bei Einrichtung eines Kommunikationskanals
  (2) Authentifizierung bei jeder Nachricht zwischen A und B
  (3) zusätzlich Verschlüsselung der Nachrichten

Kerberos-Anwendungsbeispiel: Einrichtung eines sicheren Kanals

- Gegenseitige Authentifizierung (via Kerberos Server)
- Verwendung eines Sitzungsschlüssels ("session key")
- $\{X, K, Ts, \Delta\}_\gamma$ heisst "Ticket"
  - Tickets kann man an andere ("vertrauenswürdige") Instanzen weitergeben



Hier: Version 4; spätere Versionen im Prinzip nur leicht unterschiedlich

$A \rightarrow S$: A, B

$S \rightarrow A$: $\{B, K, Ts, \Delta, \{A, K, Ts, \Delta\}_\beta\}_\alpha$

$A \rightarrow B$: $\{A, K, Ts, \Delta\}_\beta, \{A, Ta\}_K$

$B \rightarrow A$: $\{Ta + 1\}_K$

Sitzungs-schlüssel

Zeit-stempel

Gültigkeitsdauer des Sitzungsschlüssels

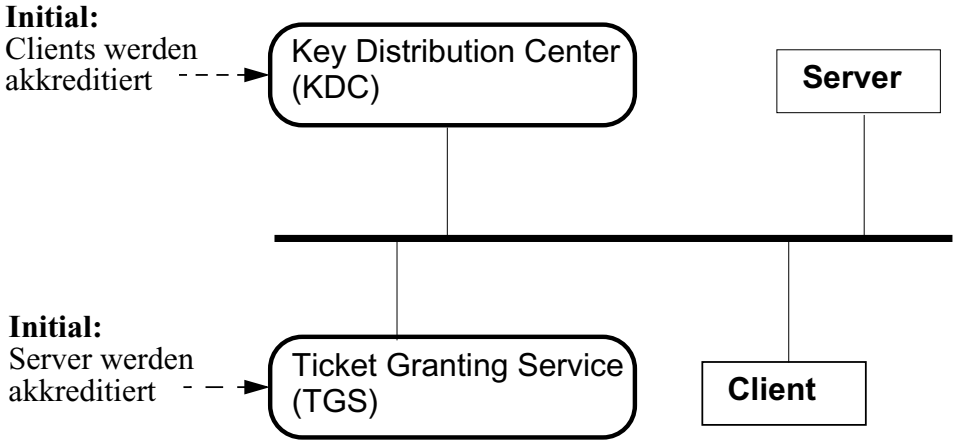
Geheime Schlüssel von A bzw. B

B sieht: A kennt K

Hiermit authentifiz. sich B gegenüber A (kein anderer konnte Ta entschlüsseln)

- Geheimschlüssel α von A und β von B darf niemand ausser S kennen! (Kenntnis wird als Identitätsnachweis betrachtet)
- A reicht hier ein von S erhaltenes (mit β codiertes) Ticket an B weiter

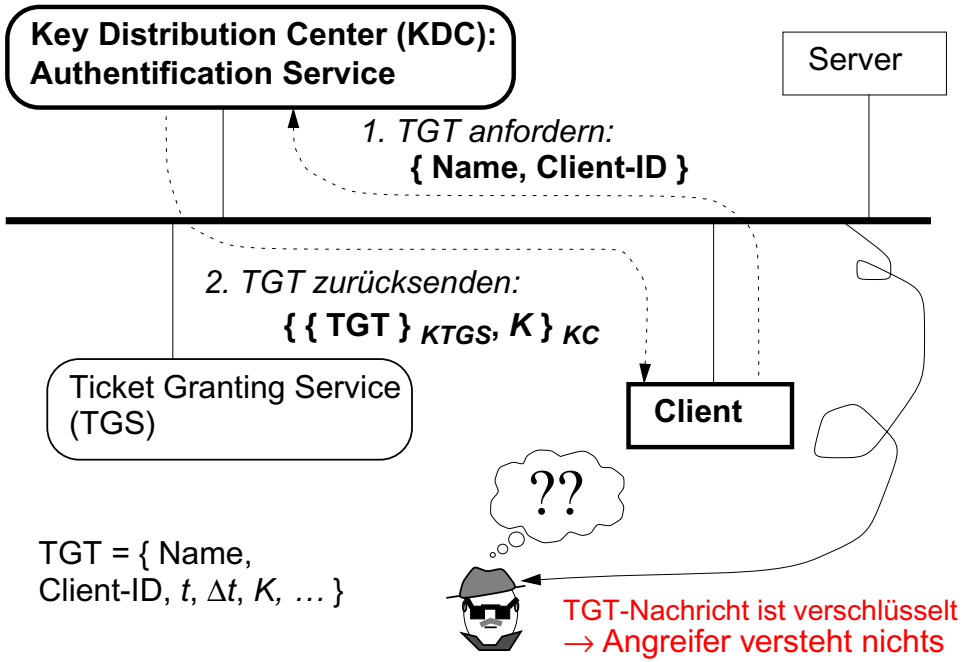
Kerberos: Akkreditierung



- Benutzer (Clients) und deren Passwörter (= Schlüssel) werden dem KDC bekannt gemacht
- TGS und dessen geheimer Schlüssel werden ebenfalls beim KDC akkreditiert
- Server und deren geheime Schlüssel werden dem TGS bekannt gemacht
  - es kann mehrere TGS-Server geben (→ Lastverteilung)

Kerberos: TGT-Anforderung

- Client erwirbt zunächst ein Ticket Granting Ticket (TGT)



- Client an KDC: sendet { Name, Client-ID } im Klartext
- KDC: wählt K; erstellt TGT = { Name, Client-ID, t, Δt, K, … }
- KDC an Client: sendet { { TGT } $_{KTGS}$, K } $_{KC}$ zurück;
  KC = h(Passwort); KTGS = TGS-Schlüssel; K = Sitzungsschlüssel
- Client: gewinnt { TGT } $_{KTGS}$ und K durch Entschlüsselung mit Passwort:
  - (chiffriertes) TGT berechtigt zum Erwerb von Service Tickets;
  - K sichert Kommunikation mit TGS gegen Angreifer

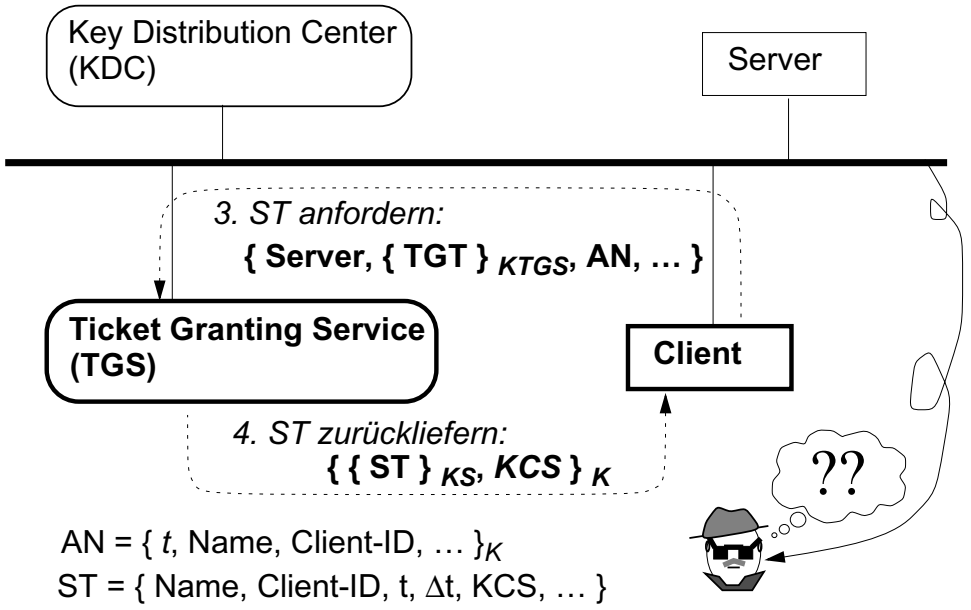
→ KDC-Nachricht ist authentisch: Nur KDC kennt noch Schlüssel KC !

→ Nur der echte Client kann TGT mittels KC nutzbar machen

→ Passwort verlässt Client-Rechner nicht

→ TGT ist verschlüsselt, nur für Zeitspanne Δt gültig, geht nur an Client

Kerberos: Service Ticket erwerben

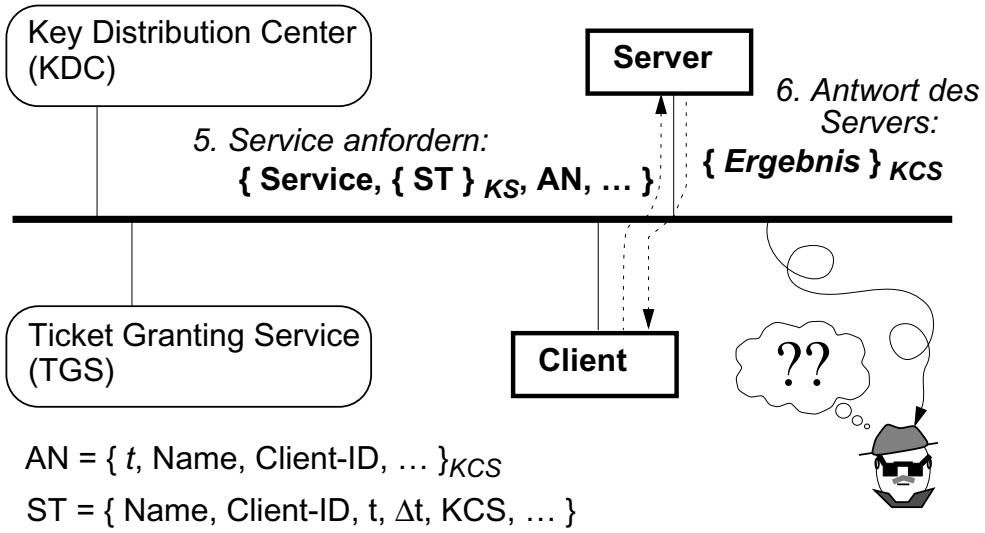


- Client: erstellt Authentizitätsnachweis AN = { t, Name, Client-ID, … } $_{K}$
- Client sendet an TGS { Server, { TGT } $_{KTGS}$, AN, … } als Request
- TGS: entschlüsselt TGT mit Schlüssel KTGS, erhält damit K;
  entschlüsselt AN mit K, vergleicht Inhalt mit TGT;
  erstellt Service Ticket ST = { Name, Client-ID, t, Δt, KCS, … }
- TGS sendet an Client { { ST } $_{KS}$, KCS } $_{K}$ zurück
- Client: gewinnt { ST } $_{KS}$ und KCS durch Entschlüsselung mit K:
  - (chiffriertes) ST berechtigt zur Nutzung des Servers
  - KCS sichert Kommunikation zwischen Client und Server

→ Ohne Sitzungsschlüssel K ist ST nicht nutzbar: Nur Client kennt K !

→ ST höchstens für Zeitspanne Δt gültig

Kerberos: Nutzung des Service



AN = { *t*, Name, Client-ID, … }$_{KCS}$

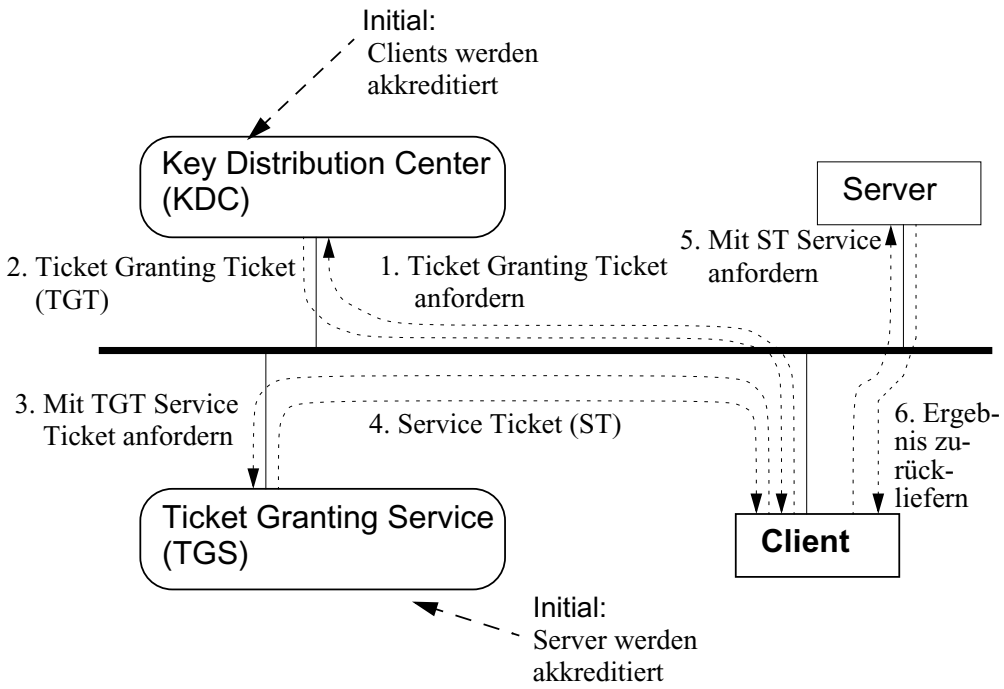
ST = { Name, Client-ID, t, Δt, KCS, … }

- Client: erstellt Authentizitätsnachweis AN = { t, Name, Client-ID, … } $_{KCS}$
- Client an Server: sendet { Service, { ST } $_{KS}$, AN, … } als Service Request
- Server: entschlüsselt ST mit KS, erhält damit KCS;
  entschlüsselt AN mit KCS, vergleicht Inhalt mit ST;
  leistet Service und erzeugt Ergebnisdaten
- Server an Client: antwortet mit { Ergebnisdaten } $_{KCS}$
- Client: authentifiziert und entschlüsselt das Ergebnis mittels KCS

→ Folgedialoge zwischen Client und Server mittels KCS verschlüsselbar

→ ST als Einmal-Ticket oder evtl. innerhalb Δt mehrfach nutzbar

Kerberos: Protokollübersicht



- Protokoll ist zweistufig:
  - Client kommuniziert nur selten mit dem KDC (1,2) → eigentlicher Geheimschlüssel (Passwort-basiert) wird nur selten benutzt
  - ein TGT ist für mehrere Anfragen beim Ticket-Service gültig

Kerberos - weitere Aspekte

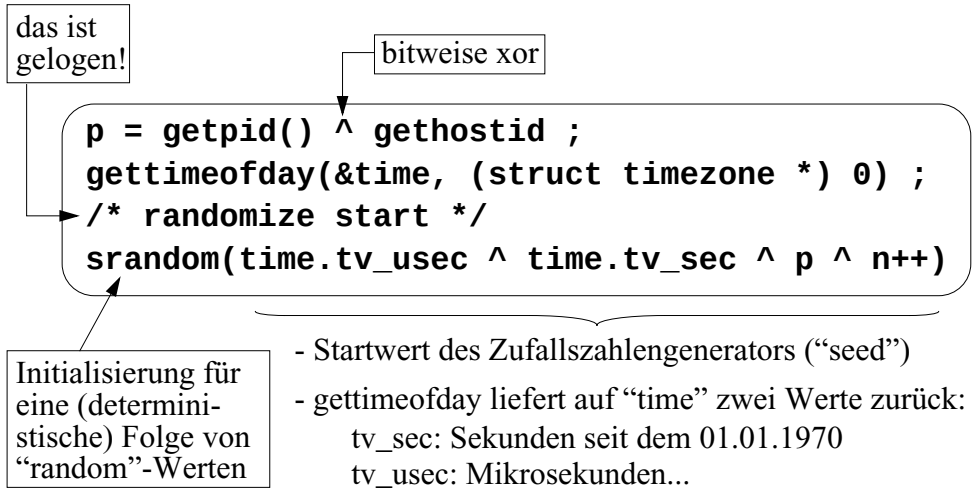
- Nachrichten enthalten noch weitere (technische) Angaben
  - z.B. Versionsnummer, Nachrichtentyp, Prüfsumme, Netzwerkadresse,...
- Es gibt dezentrale Zuständigkeitsbereiche (“realms”)
  - lok. KDC vermittelt Zugangsticket zu KDC eines fremden Bereichs
- Kerberos-Software enthält u.a.:
  - Library mit Routinen, um Authentifizierungsanforderungen erzeugen und lesen zu können, Nachrichten zu authentifizieren und zu verschlüsseln
  - Datenbank und Verwaltungsroutinen für registrierte Nutzer (Geheimschlüssel, Gültigkeitsdauer, Verwaltungsdaten,...)
  - Tools zur Replikation der Datenbank (Verteilung ist wichtig, da bei Ausfall des KDC fast nichts mehr im ganzen Netz geht!)
- Neuere Versionen (gegenüber Version 4): mehr Funktionalität und allgemeiner verwendbar, z.B.:
  - standardisierte Datenformate
  - Verbesserung einiger Sicherheitskonzepte; Alternativen zu DES
  - besser skalierbare “Cross Realm”-Authentifikation
  - Unterstützung erneuerbarer und transferierbarer Tickets
- Weiterentwicklungen
  - z.B. asymm. Schlüssel, Einbindung von Chipkarten, verteilte Datenbank,...
- Kerberos ist weit verbreitet (“Quasi-Standard”)
  - z.B. um verteilte Dateiserver zu sichern oder modifizierte Versionen von telnet, rlogin, rcp, rsh, ftp etc. zu ermöglichen
  - kommerzielle Varianten z.T. nicht kompatibel zueinander
  - Microsoft: Unterstützung ab Windows 2000

Kerberos - Sicherheitsaspekte

- KDC und TGS müssen geschützt werden
  - z.B. gegen unbefugtes Lesen der Datenbank, Verändern der Daten, denial of service,...
- Tickets müssen vom Client in einem “sicheren Speicherbereich” aufbewahrt werden
  - Master key (aus Passworteingabe des Benutzers abgeleitet) wird sobald wie möglich aus dem Speicher gelöscht
- Uhren der Kommunikationspartner und der Kerberos-Server müssen “verlässlich” synchronisiert werden
  - innerhalb eines gewissen Toleranzintervalls von einigen Minuten
  - Störung des Uhrenabgleichs erlaubt evtl. mehrfachen Ticketmissbrauch
- Replays sind innerhalb der Gültigkeitsdauer (typw.: einige Minuten bis Stunden) prinzipiell möglich!
  - Server sollte alte, noch gültige Tickets speichern, um Replays erkennen zu können
- Auf Public-Domain-Servern (und CDs etc.) könnte gefälschte Software vorhanden sein (“trojanische Pferde”)
- “Erster” Schlüssel basiert auf einem Passwort → Off-line-Attacke durch Raten gängiger Passworte
- Hintertüren ausserhalb von Kerberos
  - fremde Tickets lesen (Netz-sniffer, Superuser-Rechte beschaffen,...)
  - “Hijacking” von TCP-Verbindungen

Schlüsselgenerierung

- Die Schlüsselgenerierung von Kerberos (z.B. für TGT oder Sitzungsschlüssel) funktionierte ursprünglich so:



- Lässt sich aus einem Schlüssel der folgende berechnen?
  - p ist eine "Konstante"
  - time of day ist (ungefähr) bekannt
  - n++ unterscheidet sich i.Allg. nur in wenigen Bits von n

- Könnte jemand vielleicht absichtlich die Uhr des Servers auf einen falschen (d.h. bekannten) Wert setzen?
  - welche Granularität hat eigentlich die Uhr?

- Der Algorithmus ist inzwischen "verbessert"...

- Und die Moral der Geschichte?

Resümee (1)

- Organisatorisches zur Vorlesung

- Einordnung der Vorlesung
- Verteilte Systeme: Begriff, Sichtweisen, Eigenschaften,...
- Motivation; Gründe für verteilte Systeme
  - Kooperation von geographisch verteilten Einheiten
  - qualitatives Wachstum des Internet
- Middleware für das Internet
- Transparenzeigenschaften (Verbergen von Verteiltheit etc.)
- Verteilte Systeme als “Verbund”
- Historische Entwicklung von Systemen und Konzepten
- Charakteristika und praktische Problembereiche verteilter Systeme

Resümee (2)

- Phänomene und konzeptionelle Probleme
  - Schnappschussproblem (inkonsistente globale Sicht)
  - Phantom-Deadlocks
  - Uhrensynchronisation
  - kausal inkonsistente Beobachtungen
  - Geheimnisaustausch über unsicheren Kanal

- Multiprozessoren (gemeinsamer Speicher)
  - Buskoppelung
  - Schaltnetzkoppelung (Crossbar, Permutationsnetze)
- Cluster-Computer (verteilter Speicher)
  - Bewertungskriterien für Verbindungstopologien
  - Hypercube (rekursives Konstruktionsprinzip)
  - Torus
  - Cube Connected Cycle
  - Zufallstopologien

Resümee (3)

- Nachrichtenkommunikation
  - Message-passing-Systeme / -Bibliotheken
  - Prioritäten von Nachrichten
  - Zuverlässigkeitsgrade
- Fehlermodelle
  - fehlerhaftes Senden / Empfangen
  - Verlust von Nachrichten
  - crash, fail-stop
  - allgemeine (“byzantinische”) Fehler
- Kommunikationsmuster
  - Mitteilung ↔ Auftrag
  - synchron ↔ asynchron
- Synchrone Kommunikation
  - Definition
  - Realisierung
  - virtuelle Gleichzeitigkeit; Gummibandtransformation
  - Blockaden und Deadlocks
- Asynchrone Kommunikation
  - Vor- / Nachteile gegenüber synchroner Kommunikation
- Synchrone Kommunikation mit asynchroner simulieren
  - Warten auf ein explizites Acknowledgement
- Asynchrone Kommunikation mit synchroner simulieren
  - Puffer(prozess!) zur Entkoppelung dazwischenschalten
- Implementierung von Pufferprozessen
  - durch Inversion der Kommunikationsbeziehung
  - durch Multithreading

Resümee (4)

- Verschiedene Kommunikationsmuster
  - no-wait-send; RPC; asyn. RPC; rendezvous
- Datagramm
- Rendezvous-Protokoll
- RPC- Implementierung
  - Parameter-Marshalling
  - Stubs
  - Transparenzproblematik
- RPC-Fehlerproblematik
  - Fehlerursachen (verlorene Nachrichten, Crash von Server / Client)
  - Gegenmassnahmen
  - Probleme (Orphans,...)
- RPC-Fehlersemantik / -klassifikation
  - maybe, at-least-once, at-most-once, exactly once
- RPC- Protokolle
- RPC- Effizienz
- RPC- Binding
- Asynchroner RPC
- Beispiel: RPC bei DCE

Resümee (5)

- Socket-Programmierschnittstelle
  - Client-Server-Beispiel in C
  - als Übungsaufgabe: Sockets in Java

- Adressierungsarten
  - 1:1, direct naming
  - m:n, mailbox
  - n:1, port
  - Kanäle

Resümee (6)

- Empfangen von Nachrichten
  - non-blocking (→ aktives Warten)
  - alternatives Empfangen (“select”)
  - selektives Empfangen
  - Zeitüberwachung
  - implizites Empfangen

- Gruppenkommunikation (Broadcast / Multicast)
  - Anwendungen
  - idealisierte Sicht
  - Fehlerproblematik
  - “best effort”
  - “reliable Broadcast” mit ACK, NACK

- Algorithmus für “reliable Broadcast”
  - Fehlermodell
  - Effizienz (Zahl von Einzelnachrichten)

Resümee (7)

- FIFO-Broadcasts
  - zwei nacheinander ausgeführte Broadcasts ein und desselben Senders erreichen alle Empfänger in dieser Reihenfolge
  - nicht stark genug, um “akausale” Beobachtungen zu verhindern
- Kausale Broadcasts
  - kausale Abhängigkeit zweier Nachrichten
  - “Causal Order”: Nachrichtenempfang “respektiert” kausale Abhängigkeit von Nachrichten (ist also “kausaltreu”)
- Atomare Broadcasts
  - logisch gleichzeitiger Empfang der Einzelnachrichten eines Broadcasts
  - Realisierung über zentralen Sequencer bzw. Token auf einem logischen Ring
- Kausal atomare Broadcasts
  - virtuelle Synchronität

Resümee (8)

- Multicast
  - Zweck
  - Adressierung von Multicast-Gruppen
- Multicast: Membership-Problem
  - atomare Änderung der Gruppenzugehörigkeit
  - Tolerieren von Prozessausfällen
- Push-Prinzip und Publish & Subscribe
- Ereigniskanäle als “Softwarebus”
- Tupelräume
  - Linda-Modell
  - JavaSpaces

- Logische Zeit
  - Raum-Zeitdiagramme, Ereignisse, Kausalrelation
  - Zeitstempel von Ereignissen
  - Uhrenbedingung (als Ordnungshomomorphismus)
- Logische Uhren von Lamport
  - Definition
  - Realisierung
  - injektive Abbildung, eindeutige Zeitpunkte

Resümee (9)

- Wechselseitiger Ausschluss (mit logischer Zeit)
  - replizierte Warteschlangen von Lamport (request, reply, ack)
  - Verfahren von Ricart / Agrawala 1981
  - Korrektheitsargumente? (Exklusivität, Deadlockfreiheit, Fairness,...)

- Namensverwaltung
  - Zweck von Namen
  - Namen und Adressen
  - Binden
  - Namenskontexte, hierarchische Namensräume
  - Aufgaben einer Namensverwaltung
  - Namensverwaltung in verteilten Systemen

- Zufall
  - Pseudozufall, “echter, physikalischer” Zufall
  - Symmetrisierungstrick von J. v. Neumann

- Nameserver
  - Replikation und Caching

Resümee (10)

- Internet Domain Name Service (DNS)
  - Namensauflösung im Internet
  - “Round-Robin”-Einträge (Lastausgleich, Fehlertoleranz)
  - resource records
  - nslookup

- Client-Server-Modell (⇔ Peer-to-Peer-Strukturen)
  - Prinzip
  - Client/Server-*Maschinen*
  - Client/Server-*Rollen*
  - Server-Farmen

- Zustandsändernde / -invariante Dienste und Server
  - idempotente und wiederholbare Aufträge
  - stateless / statefull
  - Beispiel Webserver (URL rewriting, cookies)

- Konkurrente Server
  - dynamische / statische Handler-Prozesse (“slaves”)

- Leichtgewichtsprozesse (“Threads of Control”) im Client-Server-Modell
  - Vorteile und Probleme bei der Verwendung

Resümee (11)

- X-Window als “klassisches” Client-Server-System
  - aber: events zur asynchronen Rückmeldung Server → Client
- Servergruppen / verteilte Server
  - Strukturen kooperierender Server
  - Server-Auswahl bei einem Lastverbund
  - Replikation von Servern (“Überlebensverbund”)
- Middleware: historischer Kurzüberblick
- Sun-RPC
  - Identifikation entfernter Prozeduren (host, Programm-, Version-, Prozedur-Nummer)
  - Registrieren eines Dienstes auf Serverseite
  - Generieren von Prozedurstubs und Serverskelett aus Schnittstellenspezif.
- Portmapper
  - Zuordnung Port / Programmnummer eines Dienstes
- Sicherheitsaspekte bei Sun-RPC
  - “UNIX flavor”: Automatisches Mitsenden von Benutzerkennung etc.
  - “Secure RPC”: Authentifizierung (mit Diffie-Hellman / DES)
- CORBA
  - CORBA-Architektur (ORB, IDL,..)
  - Object Services
  - Erweiterungen gegenüber CORBA 2.0
  - Schicksal der Weiterentwicklung

Resümee (12)

- Middleware
  - DCOM
  - .NET-Framework
  - Web Services
  - Objektserialisierung (bei Java RMI und .NET)
- Jini
  - Motivation: Dienstparadigma, Netzzentrierung,...
  - Java-Bezug
  - Lookup-Service
  - Discovery
  - Join
  - Proxies und smart Proxies
  - Code-Mobilität
  - Leases
  - verteilte Ereignisse
  - Vorteile und Probleme von Jini

Resümee (13)

- Sicherheit in verteilten Systemen: Anforderungen
- Einmalpasswörter mit Einwegfunktionen
- One-time-Pads mit XOR
- Symmetrische und asymmetrische Kryptosysteme
- Authentifizierung mit (a)symmetrischen Schlüsseln
- Problem der “Replays” und Lösungsansätze
- Schlüsselvergabe durch Key-Server
- Autonome, “geheime” Schlüsselgenerierung
  - Schlüsselaustausch mit Diffie-Hellman-Prinzip
- “Man in the middle”: Erkennungsmöglichkeit
- Authentifizierung mit geheimen Zertifikaten
- Zero-Knowledge-Proofs
  - Beispiel: Isomorphie von Graphen

Resümee (14)

- Kerberos
  - Protokoll für Ticket-Granting-Ticket- und Service-Ticket-Erwerb
  - Anwendungsbeispiel: Einrichtung sicherer Kanäle
  - Sicherheitsaspekte