Dienste

- Namensdienst (DNS, DCE Directory Service)
- Verzeichnisdienst (LDAP)
- File-Dienst (NFS, DFS)
- Transaktionsdienst (in Datenbank-Vorlesung)
- Zeitdienst (NTP)
- Sicherheitsdienst (SSL)

Namensdienst

erlaubt Zugriff auf Ressourcen über systemweite Namen anstatt über physikalische Adressen → Auflistung mit Zuordnung zwischen Namen und Attributen von Benutzern, Computern, Diensten, entfernten Objekten, usw.

Unterscheidung:

- **Namensdienst:** entspricht Telefonbuch → Name muss für Zugriff bekannt sein
- **Verzeichnisdienst:** entspricht Gelben Seiten → Name ist nicht bekannt, aber gesuchter Benutzer, Computer, Dienst, usw. kann durch Metainformationen beschrieben werden

Namensdienst (2)

Beispiele für Namensdienste:

- DNS: Auflösung von Internet-Adressen
- RMIregistry bei JavaRMI, CORBA Namensdienst

Beispiele für Verzeichnisdienste:

- X.500 Directory Service: Standardisierter Dienst für Zugriff auf Einheiten aus realer Welt, häufig auch für Informationen bzgl. HW/SW-Dienste
- CORBA Traderdienst: Attributbasierte Auffindung von Diensten
- LDAP (Lightweight Directory Access Protocol): Sicherer (durch Authentifizierung) Zugang für unterschiedliche Internet-Verzeichnisdienste

Namensdienst (3)

Vorgänge:

- **Adressierung:** Bestimmen einer Position/Zugangsorts
- **Bindung:** Zuordnung Adresse zu Name
- **Auflösung:** Namen in zugehörige Adresse überführen

Abbildung von Namen auf Attributen des entsprechenden Objekts, nicht auf Implementierung des Objekts. Beispiele:

- DNS: Rechnername auf Attribut IP-Adresse abbilden
- CORBA Namensdienst: Abbildung Name auf entfernte Objektreferenz
- X.500: Personenname wird z.B. auf Wohnort, Telefonnummer abgebildet

Namensdienst (4)

Anforderungen an Namensdienste:

- **Skalierbarkeit:** keine a-priori Einschränkung der max. Anzahl aufzunehmender Einträge
- **Flexibilität:** Möglichkeiten für spätere Veränderungen berücksichtigen (für längere Lebensdauer sorgen)
- **Verfügbarkeit:** ohne Namensdienste fallen viele verteilte Systeme aus → Mindestverfügbarkeit sicherstellen
- **Fehlerisolation:** Fehler einer Komponente darf andere Komponenten gleicher Funktionalität nicht beeinflussen
- **Misstrauenstoleranz:** vermeide Komponenten, denen alle Benutzer des verteilten Systems vertrauen müssen

Namensräume

Namensraum: Menge gültiger Adressen, die Dienst erkennt

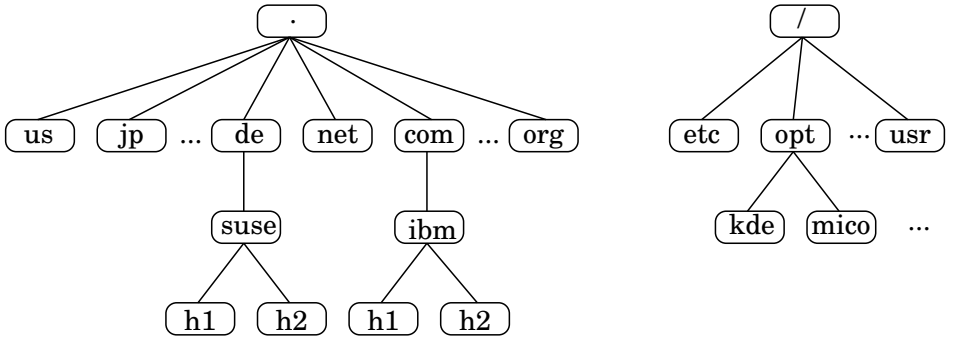
- Syntaktische Definition der internen Namensstruktur
- Name ist gültig, auch wenn es nicht gebunden ist
- Namen sollen absolut (ausgehend von der Wurzel) oder relativ (abhängig vom aktuellen Kontext) aufgelöst werden können. Beispiele:
  * Datei: `../src/prog.c` (relativ)
  * DNS: `ssh endor`, die Endung `hsnr.de` wird, bei entsprechender Konfiguration, automatisch hinzugefügt
- Aliase: Abbilden mehrerer Namen auf gleiche Adresse

Namensräume (2)

Unterscheidung

- **flacher Namensraum:** Länge des Namens durch max. Anzahl möglicher Zeichen festgelegt
- **hierarchischer Namensraum:** Anhängen von Suffixen
  * Dateisystem: `vollständiger Name = Pfad + Name`
  * DNS: `www.hsnr.de`

Namensräume (3)



Namensauflösung

Namensauflösung: einen Namen wiederholt Namenskontexten präsentieren → Navigation in Namensdiensten

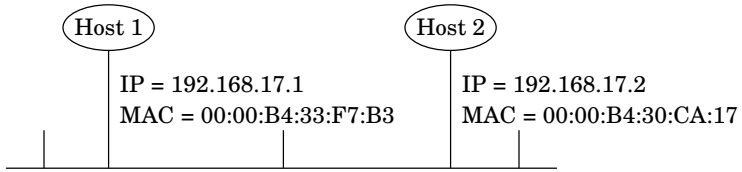
Unterscheidung abhängig von Aufteilung der Verantwortung zwischen verschiedenen Namensdiensten:

- **iterative Navigation:** der Client fragt eine Reihe von Servern ab
- **iterative, server-gesteuerte Navigation:** lokaler Server führt iterative Navigation durch, koordiniert die Anfragen und liefert das Ergebnis an Client zurück
- **rekursive, server-gesteuerte Navigation:** falls erster Server Namen nicht auflösen kann, übergeordneten Server mit Auflösung beauftragen → Server wird zum Client

Namensauflösung: ARP

Ethernet basiert auf ARP (Address Resolution Protocol):

- jede Netzwerkkarte hat weltweit eindeutige, fest eingebrannte Adresse (Media Access Control, MAC)
- Zugriff auf Netzwerk wird über MAC-Adresse kontrolliert: die Knoten werden grundsätzlich mit ihrer MAC-Adresse angesprochen, nicht mit der IP-Adresse.
- nicht auf IP basierende Netze: Novell NetBIOS, Microsoft NetBEUI, Apple Appletalk, ...



Namensauflösung: ARP (2)

ARP arbeitet in drei Schritten:

1. Host 1 schickt Nachricht an alle Rechner: eigene IP- und MAC-Adresse, IP-Adresse des gesuchten Hosts
2. Rechner des Netzwerks speichern IP- und MAC-Adresse des Absenders für mögliche, spätere Verbindungen
3. Host 2 sendet MAC-Adresse seiner Netzwerkkarte

⇒ Host 1 kennt MAC-Adresse und baut Verbindung auf

Falls Host 2 nur über Router von Host 1 erreichbar: Host 1 erfragt MAC-Adresse des Routers, Host 1 übergibt Router die Daten, Router kümmert sich um weitere Zustellung.

Zuordnung MAC- zu IP-Adresse: gespeichert in Tabelle, unter Linux einsehbar mittels `arp`-Kommando

DARPA Internet Domain Name Service

Haupteinsatzzweck von DNS:

- Auflösung von Hostnamen/Umwandlung in zugehörige IP-Adresse
- Mailhost-Position: welcher Mailserver ist für Empfang bzw. Auslieferung von E-Mails zuständig

Spezielle Implementierungen verfügen über:

- Umgekehrte Auflösung: ermitteln des Hostnamens anhand IP-Adresse, z.B. `nslookup`
- Host-Informationen: Angaben über aktuelle Architektur, Maschinentyp, Betriebssystem, ... (Vorsicht: Infos können für Angriffe missbraucht werden)

DNS-Abfragen

- Skalierbarkeit von DNS durch Partitionierung, sowie Replikation und Caching der einzelnen Partitionen → Aufteilung über ein logisches Netzwerk von Servern
- Aufteilung der DNS-Datenbank in Zonen definiert durch
  * Attributdaten für Namen in einer Domain minus aller Subdomains, die von tiefer liegenden Autoritäten verwaltet werden
  * Verweise auf mind. zwei übergeordnete Name-Server
  * Verweise auf DNS-Server für untergeordnete Domains
- Replikation: primärer und mind. ein sekundärer Server
  * primärer Server liest die Daten direkt aus einer Datei
  * Slaves laden Daten regelmäßig vom Master herunter oder werden vom Master bei Änderungen mit Daten versorgt (Zonentransfer)

DNS-Auflösung

Verteilung und Lokation der Name-Server ist für Clients transparent durch Verwendung von Agenten (genannt Auflöser oder Resolver)

- üblicherweise Bestandteil der Standardbibliotheken
- Anfragen werden angenommen, entsprechend dem DNS-Protokoll verpackt und versendet
- einfaches Request/Reply Protokoll (mehrere Anfragen können in einem Paket versendet und entsprechend viele Antworten empfangen werden)
- Spezifikation der Vorgehensweise (iterativ, iterativ Server-basiert, rekursiv)

DNS: Implementierung

bekannt: BIND (Berkeley Internet Name Domain)

- Server führen `named`-Dämon aus und hören festgelegte Portnummer ab (Konfigurationsdatei `/etc/named.conf`)
- Clients binden Resolver aus Standardbibliothek ein (`gethostbyname`, `gethostbyaddr`)
- Unterstützung von Primär-Server, Sekundär-Server und reine Caching-Server
- Veraltete Einträge werden nicht erkannt: ändern sich Namensdaten, liefern Server evtl. noch mehrere Tage später alte Daten
- Caching unter Linux mittels des Name Service Caching Daemon `nscd` (Konfigurationsdatei `/etc/nscd.conf`)

DNS: Implementierung (2)

Jeder Datensatz einer DNS-Datenbank besteht aus folgenden Elementen:

- Domain-name: Name der Domain, auf den sich der Datensatz bezieht
- Time-to-live: wie lange soll ein Eintrag gültig sein? (ist optional, wird oft weggelassen)
- Class: für Internet-Infos steht hier immer `IN`
- Value: Wert des Datensatzes, abhängig vom Typ

DNS: Implementierung (3)

- Type: um welchen Typ Datensatz handelt es sich?
  * SOA (Start Of Authority): verschiedene Parameter der Zone, die Name-Server verwalten soll
  * A (Address): Adresse eines Internet Hosts
  * MX (Mail Exchange): Priorität und Name des Mail-Servers der Domain
  * NS (Name Server): Name-Server der Domain
  * CNAME (Canonical Name): Domain-Name eines Rechners (Aliasfunktion)
  * PTR (Pointer): Alias für eine numerische IP-Adresse
  * HINFO (Host Information): ASCII Beschreibung des Hosts (CPU, OS, ...)
  * TXT (Text): nicht verwertbarer Text - Kommentar

DNS: Implementierung (4)

DNS-Server: Konfigurationsdatei `/etc/named.conf`

- Angabe des Verzeichnisses, wo weitere Info-dateien des Name-Servers liegen
- Definition der Zonen, die Name-Server bedient
- Eine Zugriffskontroll-Liste (Access Control List - ACL) wird mit dem acl-Befehl erstellt:

```
acl my_net {10.230.1.0/24; 10.230.4.0/24; };
```

⇒ Name-Server nur für lokales Netz zugreifbar! Server, die fremden Rechnern Informationen über interne Adressen anbieten, dürfen keine Einschränkungen haben.

DNS: Implementierung (5)

Zone, die Verbindung zu Root-Servern herstellen kann:

```
zone "." in {
        type hint;
        file "root.hint";
};
```

zwei Zonen, die grundsätzlich vorhanden sein müssen:

```
zone "localhost" in {
        type master;
        file "localhost.zone";
};
zone "0.0.127.in-addr.arpa" in {
        type master;
        file "127.0.0.zone";
};
```


DNS: Implementierung (6)

Beispiel: DNS-Server für ein Netz `10.230.1.0/24` mit dem Domain-Namen `mydomain.de`

```
zone "mydomain.de" IN {
    type master;
    file "mydomain.de.zone";
};
zone "1.230.10.in-addr.arpa" in {
    type master;
    file "10.230.1.zone";
};
```

DNS: Implementierung (7)

DNS-Client: drei Konfigurationsdateien unter `/etc`

- `host.conf`: contains configuration information specific to the resolver library
- `nsswitch.conf`: (ab `glibc2`) System Databases and Name Service Switch configuration file
- `resolv.conf`: contains information that is read by the resolver routines the first time they are invoked by a process. The file is designed to be human readable ...

```
search kr.hs-niederrhein.de hs-niederrhein.de
nameserver 194.94.120.1
nameserver 194.94.120.2
```


DCE Directory Service

besteht aus zwei Komponenten:

- **Cell Directory Service (CDS):** verwaltet Namensraum einer Zelle, jede Zelle besitzt mind. einen CDS-Server
- **Global Directory Service (GDS):** verwaltet globalen Namensraum außerhalb und zwischen den Zellen, verbindet Zellen untereinander

Namensnotation:

- weltweite Struktur mit globaler Wurzel `/...`
- Name einer Zelle wird in X.500-Notation oder in DNS-Notation angegeben
- jede Zelle besitzt eigenen, internen Namensraum

DCE Directory Service (2)

DCE-Name besteht aus drei Teilen:

- **Präfix:** `/...` steht für global, `/.:` steht für lokal
- **Zellname:** muss nur bei globalen Namen angegeben werden
- **lokaler Name:** identifiziert ein Objekt innerhalb einer Zelle, Notation nach Unix File-Benennungsschema
- die einzelnen Teile werden mittels Schrägstrich getrennt

X.500-Notation:

- hierarchisches, attribut-basiertes Namensschema
- ISO/OSI-Standard
- `Country=DE/OrgType=EDU/OrgName=HSNR/Dept=FB03/...`

Cell Directory Server

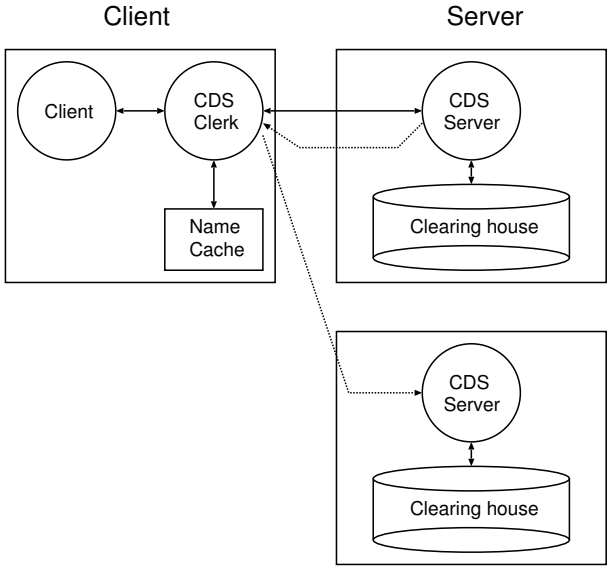
verwaltet Verzeichnisse einer Zelle:

- Verzeichniseintrag besteht aus Name und Attributen: Schutzinformationen/Zugriffsrechte, Lokation, Charakteristiken (z.B. bei Drucker: Typ und Geschwindigkeit)
- **Clearing house:**
  * Datenbank, wird vom CDS-Server verwaltet, enthält Verzeichnisse
  * ein Server kann mehrere Datenbanken verwalten
  * Wurzelverzeichnis ist in allen DBs repliziert → Suche kann bei irgendeinem CDS-Server begonnen werden

Cell Directory Server (2)

- **Clerks:**
  * Agent (Transparenz bzgl. replizierter Daten)
  * läuft auf jeder Maschine, die einen CDS-Server benutzt
  * erhält Anfragen eines Clients, befragt einen/iterativ mehrere CDS-Server, stellt Antwort zusammen und gibt sie an Client zurück
  * verwendet Cache, der regelmäßig auf Platte geschrieben wird (überlebt Neustart des Systems/Anwendung)
- **Lookup:** Namensresolution/-auflösung

Cell Directory Server (3)



Global Directory Server

Auffinden eines CDS-Servers in einer anderen Zelle mittels Agententechnologie: **Global Directory Agent**

- GDA kann auf beliebiger Maschine liegen (unabhängig von CDS)
- erhöhen der Verfügbarkeit und Zuverlässigkeit: jede Zelle kann mehrere GDAs besitzen
- jeder CDS-Server einer Zelle besitzt Information über Lokation des lokalen GDA

Verzeichnisdienst
LDAP: Lightweight Directory Access Protocol

- im Prinzip eine hierarchisch strukturierte Datenbank, in der beliebige Informationen abgelegt werden können
- oberste Ebene enthält immer das Element `root` (Wurzel des Verzeichnisdienstes, analog zur Wurzel / des Verzeichnisbaums in Unix-File-Systemen)
- fachlicher Zusammenhang zwischen Dateisystem und dem Verzeichnisdienst besteht nicht
- strukturierte Speicherung der Informationen → schneller Zugriff auf Daten (es muss nur in einem Teil des Baums gesucht werden)
- Organisation des Verzeichnisdienstes und Bearbeiten der enthaltenen Informationen erfolgt über LDAP

LDAP (2)

Begriffe:

- **Objekt:** Eintrag im Verzeichnisdienst. Objekt root ist grundsätzlich vorhanden, muss nicht explizit angelegt werden, beschreibt Anfang der Struktur
- **Container:** Objekte, die nur strukturelle Funktion haben. Enthalten keine Informationen, sondern dienen der Gliederung. (analog: Verzeichnisse in Dateisystemen)
- **Blattobjekte:** enthalten die eigentlichen Informationen (analog: Dateien in Dateisystemen)

LDAP (3)

Vordefinierte Container-Objekte:

- root: Wurzel des Baums
- country: beschreibt ein Land, Kürzel `c`, kann nur unterhalb von root erstellt werden
- organization: ist ein country-Objekt vorhanden, so muss ein Organization-Objekt direkt unterhalb eines Landes stehen, sonst steht organization direkt unter root. Bezeichnung durch `o`.
- organizational unit: Organisation kann in verschiedene Units gegliedert sein, Kennzeichnung durch `ou`, kann auch unterhalb von `ou` stehen
- common name: eigentliche Informationen ist in Blattobjekten enthalten, Kürzel `cn`

LDAP (4)

Prinzip der Replizierung:

- auf beiden Servern (Master und Replica): LDAP-Server installieren und Daten einmalig synchronisieren
- Konfigurationsdatei `slapd.conf` des Masters: Parameter `replogfile` und `replica` eintragen → im `replogfile` werden alle Änderungen am Datenbestand protokolliert, Master schickt diese Datei selbständig dem Replica zu, Replica führt diese Änderungen auf eigenem Datenbestand nach
- Parameter `updatedn` in Konfigurationsdatei des Replica-Rechners eintragen → definiert das Objekt, das Update-Informationen einpflegen kann

LDAP (5)

Prinzip der Replizierung: (Fortsetzung)

- Datenkonsistenz: durch Vergabe von access-Regeln nur lesenden Zugriff erlauben
- auf Master den `slurpd` starten → `replogfile` periodisch zur Replica übertragen

sonstiges:

- LDAP Data Interchange Format (LDIF)
- Kommandos: `ldapadd`, `ldapsearch`, `ldapmodify`
- Client-Konfigurationsdatei: `ldap.conf`

LDAP (6)

Auszug einer Konfigurationsdatei `slapd.conf`:

```
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/inetorgperson.schema
database        ldbm
suffix          "o=ibm"
suffix          "dc=ldap,dc=ibm,dc=com"
rootdn          "cn=Admin,o=ibm"
rootpw          secret1
directory       /var/lib/ldap/ibm
...
```

to be continued ...

File-Dienst

Verteilte Dateisysteme unterstützen gemeinsame Nutzung von Dateien im Intranet:

- Zugriff auf Dateien unabhängig von deren tatsächlicher Position
- Grundlage für effiziente Umsetzung zahlreicher Dienste wie Namens-, Druck-, Authentifizierungsdienste, Web-Server, ...
- Beispiele: NFS (Network File System), AFS (Andrew File System) und deren kommerzielle Implementierung DFS (DCE Distributed File System)

File-Dienst (2)

Anforderungen an verteilte Dateisysteme:

- Dateisystem ist oft der am stärksten ausgelastete Dienst im Intranet → Funktionalität und Leistung kritisch für Gesamtsystem
- **Zugriffstransparenz:** keine Unterscheidung zwischen lokalen und entfernten Daten
- **Ortstransparenz:** Clients sehen einheitlichen Namensraum → Dateien und Dateigruppen können an andere Position verschoben werden, ohne dass sich ihre Pfadnamen ändern
- **Mobilitätstransparenz:** weder Clients noch Administrationstabellen in Clients werden bei Datenverschiebung modifiziert

File-Dienst (3)

Anforderungen an verteilte Dateisysteme: (Fortsetzung)

- **Leistungstransparenz:** eine Mindestverfügbarkeit auch bei sehr hoher Auslastung sicherstellen
- **Skalierungstransparenz:** einfache inkrementelle Erweiterung des Dateidienstes möglich
- **Offenheit:** heterogene Hardware und Betriebssysteme werden unterstützt
- **Nebenläufige Dateiaktualisierung:** Dateiänderungen sollen keine Operationen der anderen Clients stören → Kontrolle für schreibende Zugriffe (z.B. UNIX-Standard `lockd`)

File-Dienst (4)

Anforderungen an verteilte Dateisysteme: (Fortsetzung)

- **Dateireplikation:** Datei wird durch mehrere Kopien an unterschiedlichen Positionen dargestellt → erhöht die Fehlertoleranz und ermöglicht eine Lastbalancierung
- **Effizienz:** verteilter Dateidienst soll bzgl. Leistung, Umfang, Funktionalität und Zuverlässigkeit mit einem lokalen Dateisystem vergleichbar oder besser sein
- **Sicherheit:** Zugriffskontrollmechanismen wie in lokalen Dateisystemen mit zusätzlichen Forderungen nach
  * Verschlüsselung und Signierung von Anforderungs- und Antwortnachrichten
  * zuverlässiges Abbilden von Benutzer-ID auf lokale ID

File-Dienst (5)

Anforderungen an verteilte Dateisysteme: (Fortsetzung)

- **Fehlertoleranz**
- **Konsistenz:** Modell für nebenläufigen Dateizugriff so, dass alle Prozesse den gleichen Inhalt sehen, als gäbe es nur eine einzige Datei

Network File System - NFS

- 1985 von Sun entwickelt, NFS-Protokoll ist ein Internet-Standard und definiert in RFC 1813
- es wird kein Dateisystem im Netz verteilt (es ist kein verteiltes Dateisystem im engeren Sinn)
- Zugriffe auf entfernte Dateien werden in Aufträge an Server umgesetzt (Dateien werden im Netzwerk zur Verfügung gestellt)
- symmetrische Client/Server-Beziehung: Rechner kann gleichzeitig Client (Dateien importieren) und Server (Dateien exportieren) sein

NFS (2)

- keine Einschränkung bezüglich der Anzahl importierter/ exportierter Dateisysteme
- zustandsloser Server:
  * Übertragung der Zugangsdaten und Sicherheitsüberprüfung bei jedem Zugriff erforderlich (GID/UID)
  * Verifikation durch Zusatzdienste wie NIS (Network Information Service)

  ⇒ bei Absturz des Servers gehen keine Informationen verloren, aber Server besitzt keine Informationen über gesetzte Sperren (locks)
- mount: Einbinden eines (Teil-) Dateisystems ins lokale Dateisystem (analog: Einbinden eines CDROM-Dateisystems)

NFS: mount

- Zuständigkeit bei Mount-Prozess im Benutzerraum:
  * Datei `/etc/exports` bei NFS-Server: welche lokalen Dateisysteme dürfen exportiert werden
  * Liste berechtigter Clients für jeden Namen angeben
- Integrationsmöglichkeiten:
  * einbinden beim Booten (`/etc/fstab`)
  * manuell einbinden mittels `mount`
  * Automounter: Verzeichnisse beim ersten Zugriff (für die Nutzungsdauer) einbinden

NFS: Protokoll

- NFS-Protokoll verwendet XDR und UDP
- NFS-Server: 15 Operationen (Prozeduren)
  * `lookup(dir, name)` liefert File-Handle und -Attribute für die Datei `name` aus dem durch `dir` spezifizierten Verzeichnis
  * `read(file, offs, cnt)` liest bis zu `cnt` Zeichen aus der Datei `file` beginnend bei `offs`
  * `write(file, offs, cnt, data)` schreibt `cnt` Zeichen in die Datei `file` beginnend bei `offs`
- keine Dateioperationen `open` und `close` (widersprechen Zustandslosigkeit des NFS Dienstes)

NFS: Protokoll (2)

- Sperren trotz Zustandslosigkeit durch zusätzliches Protokoll mit `lockd` (Lock-Dämon)
  * verwaltet Tabellen zur Speicherung von Sperren auf lokalen Dateien
  * auf jeder Maschine läuft ein `lockd`: behandelt Locks auf lokalen Dateien oder sendet die Lock-Operation an den `lockd` eines anderen Rechners
- NFS unterscheidet Lese- und Schreibsperren
- im Fehlerfall kompliziertes Freigeben/Weiterverwenden der Sperren mittels Statusmonitor `statd`
- Blockgröße bei Transfer: 8K Byte
- NFS verwendet Cache fester Größe

DCE Distributed File System - DFS

- echtes verteiltes Dateisystem
- Blockgröße bei Transfer: 64K Byte
- DFS-Server ist multithreaded und zustandsspeichernd
- DFS-Client verwendet Cache
- DFS bietet Unix-File-Semantik: jede Leseoperation sieht Effekte von vorher durchgeführten Schreiboperationen auf der Datei (realisiert mittels Token-Manager)
- DFS unterstützt File-Replikation

DFS: Token-Manager

Performancesteigerung durch

- verwenden **operationsspezifischer Token** für Open-, Read-, Write-, Lock- und Status-Operationen (Check-File und Update-File)
- **fein granulierte Token:** beschränken Read-, Write- und Lock-Operationen auf kleinen Teil einer Datei

umgehen von Client-Ausfällen:

- Client reagiert nicht auf Token-Entzugsaufforderung: Server wartet 2 Minuten
- nach Ablauf der Frist: Server nimmt an, dass Client ausgefallen ist und verwendet neues Token

DFS: File-Replikation

zur Erhöhung der Zuverlässigkeit/Verfügbarkeit und des Systemdurchsatzes werden Dateien repliziert

unterscheide drei Vorgehensweisen:

- **explizite Replikation:** Programmierer legt Kopien an und verwaltet sie
- **Master/Slave-Server:** · Server legt Datei an und kopiert sie auf die Slaves · Änderungen erfolgen zunächst auf dem Server, Server teilt Änderungen den Slaves mit · Single-Point-of-Failure
- **File Suite:** · erfordert einen verteilten Mechanismus: Änderungskonflikte auflösen und Konsistenz der Kopien sicherstellen · verwende Zeitstempel pro Kopie, bei Zugriffen nur aktuelle Kopie verwenden

Zeitdienst

Ziel: max. Abweichung der Uhren aller Rechner minimieren

oft: ein Rechner R besitzt Funkempfänger für UTC-Signal $\rightarrow$ synchronisiere alle anderen Rechner mit R

erster Versuch: (Synchronisation an UTC-Rechner)

- zwei Uhren können nach Zeit t um $2p \cdot t$ Sek. abweichen (p: max. Abweichung, wird vom Hersteller angegeben) $\rightarrow$ Uhren alle $d/2p$ Sekunden synchronisieren
- jeder Client erfragt regelmäßig beim Zeit-Server die aktuelle Zeit und setzt seine Uhr entsprechend

Probleme: · Zeit darf niemals rückwärts laufen · Nachrichtenlaufzeit der Antwort verfälscht Synchronisation

Zeitdienst (2)

Lösung:

- zu schnelle Uhren verlangsamen: vom Timer ausgelöste Unterbrechungsroutine addiert weniger Millisekunden auf als üblich, bis Angleichung vollzogen ist
- messen der Verzögerungszeit/schätzen der Dauer der Unterbrechungsbehandlung $\rightarrow$ einfache Übertragungszeit

$$T = (T_1 - T_0 - T_{interrupt})/2$$

Zeitdienst (3)

anderer Ansatz (BSD-Unix):

- Zeit-Server fragt periodisch die Rechner nach ihrer Zeit
- aus den Antworten berechnet er eine mittlere Zeit
- Server teilt Rechnern neue Zeit mit: Rechner setzen ihre Uhr auf die neue Zeit oder verlangsamen ihre Uhr

verteilter Algorithmus:

- Zeit einteilen in Synchronisationsintervalle fester Länge
- zu Beginn eines Intervalls: lokale Zeit an alle anderen Rechner schicken
- neue Zeit berechnen aus allen, innerhalb eines gegebenen Zeitintervalls eintreffenden Nachrichten

NTP: Network Time Protocol

Eigenschaften:

- präzise Synchronisation eines Clients mit UTC trotz großer/variabler Nachrichtenverzögerungen bei Kommunikation in großen Netzen (Internet)
- redundante Server und Pfade ermöglichen zuverlässigen Dienst (übersteht Verbindungsunterbrechungen)
- Rekonfiguration der Server bei Ausfall eines Servers
- Auslegung auf große Anzahl Clients und Server: häufige Synchronisation $\rightarrow$ ausreichende Anpassung der Abweichgeschwindigkeit
- Authentifizierungsschutz vor Manipulation

NTP (2)

Architektur:

- Zeit-Server im hierarchischen baumartigen Subnetz
- primäre Server in Wurzelschicht besitzen UTC-Empfänger
- sekundäre Server in Schicht 2 werden direkt vom primären Server synchronisiert
- Genauigkeit nimmt zu den Blättern hin ab
- NTP berücksichtigt Gesamtverzögerungen: bewerte Qualität der Zeitdaten auf einem bestimmten Server
- Infos unter `http://www.ntp.org`

NTP (3)

Synchronisationsmöglichkeiten:

- Multicast: Verwendung bei schnellen lokalen Netzen
  * Server schickt periodisch aktuelle Zeit per Multicast an LAN-Rechner
  * geringe Genauigkeit
- Procedure-Call-Modus:
  * Server beantwortet Zeitanfragen mit Zeitstempel
  * mittlere Genauigkeit
- symmetrisch: Synchronisation zwischen Zeitservern
  * wechselseitiger Austausch von Zeitstempeln
  * hohe Genauigkeit

NTP: Implementierung

Konfigurationsdatei: `/etc/ntp.conf`

```
## addresses of Radio and modem clocks: 127.127.t.u
## t: clock type, u: unit number in the range 0-3
## server 127.127.8.0 mode 5 prefer

## fake driver (when no synchronized time is available)
## server 127.127.1.0         # local clock (LCL)

server  time.hsnr.de          # name of server
server  bla.bla.de            # name of server

driftfile  /etc/ntp.drift     # path for drift file
logfile    /var/log/ntp       # alternate log file
```