Namensdienst für SDI-OS

SDI4

Martin Riedlberger
Tobias Modschiedler

Motivation

- Wie wird Dienst xy gefunden?
- Wie können Dateien benutzt werden?
- Woher weiß ich die Thread-ID von yz?
- …

(wenn der Name jeweils bekannt ist)

Anforderungen

- hierarchische Namensräume
- verteilte Auflösung
- Interfaces zur Namensauflösung, Verwaltung, Enumeration
- Quellnamensraum: menschenlesbare Strings, Wurzel- und Trennzeichen: /

Umsetzung

- Root-NS mit fester Thread-ID (Closure)
- Nameserver speichert Einträge in Katalogen
  - Root-Katalog implizit
- es gibt 3 Arten von Einträgen:
  - Nameserver, Handle, Catalog

Umsetzung

- Anfrage: String mit gesuchtem Pfad
- Rückgabe: SDI_nsentry
  - unsigned entrytype
  - L4_ThreadId_t nextserver
  - L4_Word_t value
    Anzahl abgearbeiteter Zeichen oder Handle
  - L4_StringItem_t path

Eintrag Typ: Nameserver

- anderer NS ist für Rest der Auflösung zuständig
- path: Rest des Pfades, mit dem der Client sich iterativ an nextserver wenden muss

Eintrag Typ: Catalog

- Verweis auf Unterkatalog des Servers
- Server kann selbst weiter auflösen

Eintrag Typ: Handle

- Auflösung fertig
- nextserver: zuständiger Server (Beispiel: Dateiserver)
- value: gültiges Handle bei nextserver für das angefragte Objekt

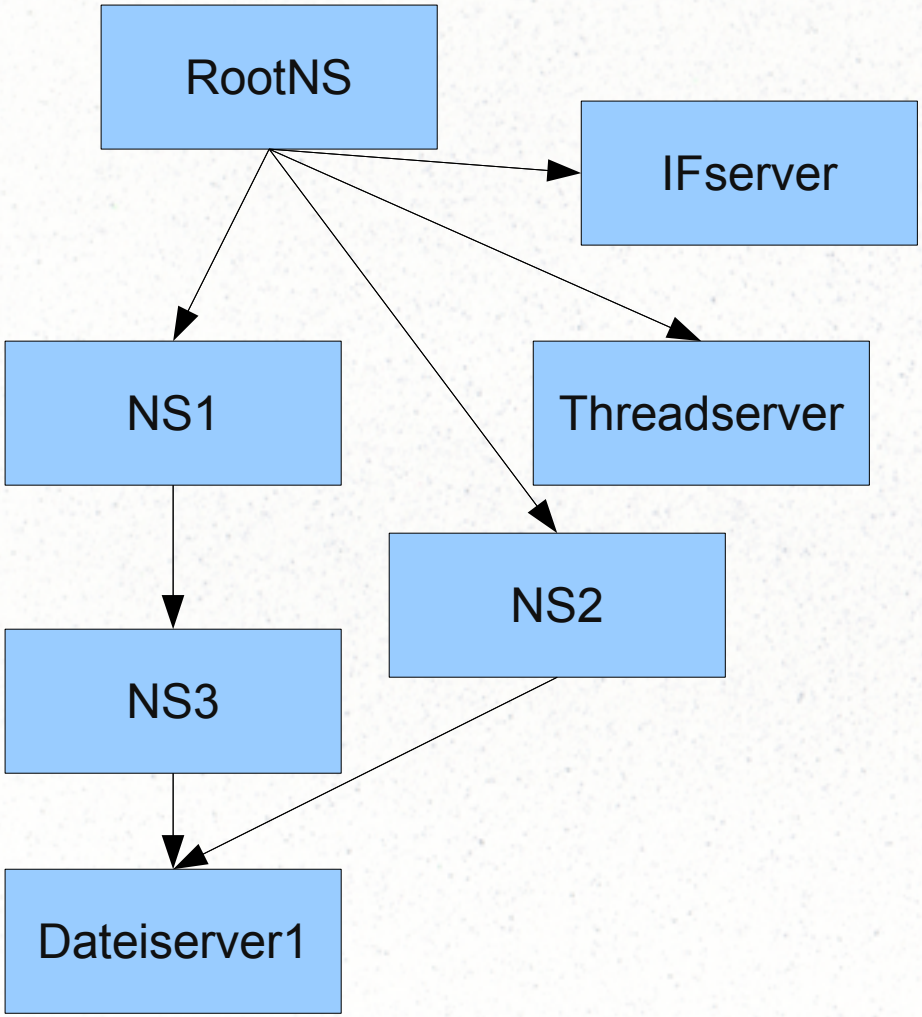
put / remove

- andere Server können/müssen ihre Objekte in NS eintragen (put)
- Stelle für Eintrag wird iterativ gesucht, beginnend beim Root-NS
- Verwendung der gleichen Datenstruktur wie bei resolve (SDI_nsentry)

Einbindung

- Objekte wie z.B. Threads oder Interfaces adressieren
- benötigt zusätzliche Server, die sich beim Root-NS unter einem Präfix registrieren (Bsp. /threads bzw. /interfaces)
- diese Server haben dann Zuordnung Handle → Objekt

Hierarchie



- Root-NS
- weitere Sub-NS
- “Objekt-Server”

Enumeration

- per Iterator
- Zustand wird im Server gespeichert
  - für Clients einfach zu benutzen
- muss wieder freigegeben werden
  - aus Sicherheitsgründen limitierte Anzahl Iteratoren pro Client

IDL4 Interfaces

- Resolver
- Binder
- Iterator

Resolver

```
interface Resolver
{
   void resolve(in string path,
                out SDI_nsentry entry)
   raises (not_found);
};
```

Binder

```
interface Binder
{
   void put(SDI_nsentry entry)
   raises (not_found);

   void remove(in string path)
   raises (not_found);
};
```

Iterator

```
interface Iterator
{
   int getIterator(in string path)
   raises (not_found);

   void next(in int id,
             out SDI_nsentry entry)
   raises (invalidId)

   void release(in int id)
   raises (invalidId);
};
```

Ausblick

- Sicherheit
  - wer darf put/remove benutzen?
- Performance
  - caching
- Konsistenz
  - remove

Ende

Gibt es Fragen zu unserem Entwurf?