Verteilte Sensornetzwerke

Mit Datenaggregation und Sensorfusion

PD Stefan Bosse

Universität Bremen - FB Mathematik und Informatik

Das Semantisch Organisierte Sensornetzwerk und Gruppenkommunikation

Wie können Sensorknoten semantisch miteinander Daten austauschen

Wie können Sich einzelne Knoten miteinander verbinden ohne IP Adressen zu nutzen?

Services vs. Geräten

Semantik von Daten vs. IP Adressen

Geospatiale Semantische URLs

- Eine URL kann neben einen Server/Service und dem verwendeten Kommunikationsprotokoll weitere Informationen über Pfade und URL Parameter enthalten:

```
URL Schema = PROTO://SERVER/PATH?PARAMTER
```

Die Idee: Pfade werden benutzt um Sensoren, d.h., Sensorknoten zu erreichen

- Dabei kann ein Sensorknoten seinen Dienst in einem Verzeichnisserver (Broker) registrieren: Directory Name Service (DNS)

- Die Pfade können aus geospatialen und sensorsemantischen Kennzeichnern zusammengesetzt werden
- Die Verzeichnise können geospatiale Räume abbilden
- Sensorknoten können in beliebig vielen Verzeichnissen registriert werden

Beispiele

```
dns/geo/bremen/sensor/temperatur/*
dns/sensor/temperatur/ba8b195b-6810-ea2e-457a-8b629382c80c
dns/sensor/temperatur/location/room/
dns/geo/bremen/sensor/nodes/*
```

Wie könnte ein geeignetes semantische Schema aussehen (Klassifiaktion)

- Jeder Sensorknoten erhält eine eindeutige Portadresse im UUID Ver. 4 Format

```
XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
ba8b195b-6810-ea2e-457a-8b629382c80c
0c9c23d4-15d0-ab81-207a-7c24dba9c00b
```

Sensor Router: Verzeichnisserver und Kommunikationsmultiplexer

- Es wird WebSocket Kommunikation zwischen einem Sensorknoten und dem öffentlich sichtbaren Sensor Router (inkl. DNS) verwendet
- Der Kommuniaktionskanal überträgt RPC Nachrichten vom Endpunkt zum Router un umgekehrt
- Der Sensor Router verwaltet die Verzeichnisse und trägt neue Sensorknoten in den Verzeichnissen ein
- Andere Sensorknoten oder Rechner können mit einem Sensorknoten über den Sensor Router (Multiplexer) verbunden werden → Virtuelle Kanäle (Virtual Channel, VC)

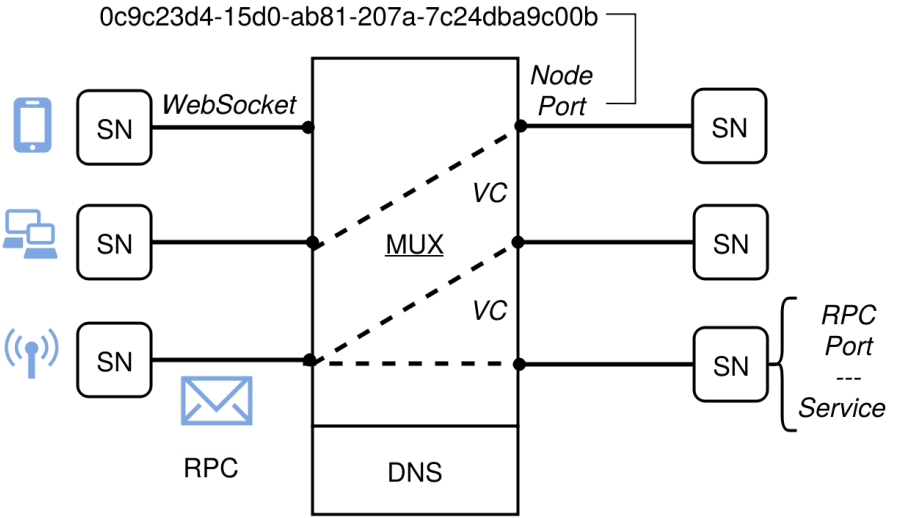


Abb. 1. Kommunikationsarchitektur des Sensor Routers mit DNS

Remote Procedure Call

Endpunkt

- Jeder Endpunkt (Sensorknoten) kann über einen WebSocket Kanal einen RPC Service zur Verfügung stellen:
  - Sensor Information (Messeinheit usw.)
  - Sensor Lesen
  - Sensor Kalibrieren
  - Sensor Schreiben (wenn möglich)
- Jedser RPC Service hat einen Port und stellt Operation via Capabilities bereit

Sensor Router

- Ebenso stellt der Sensor Router auf der anderen Seite einen RPC Service zur Vefrügung:
  - Eintragung des Sensorknoten (bzw. dessen Kommunikationsports) in ein Verzeichnis im DNS
  - Erzeugung oder Entferneung von Verzeichnissen im DNS
  - Suche von Kommunikationspartnern über DNS Pfadsuche (ggfs. mit RegEx Mustern und Wildcards)
  - Verbindung mit anderen Kommunikationsendpunkten über Virtuelle Kanäle (VC)

Zugriffsrechte und Authorisierung

Bisher könnte jeder Sensorknoten der mit dem Sensor Router eine Verbindung hat alle Operationen ausgühren

- Sogenannte Capabilities geben dem Klienten (Endpunkt) bestimmte Rechte für Operationen
- Eine Capability besteht aus:
  - Serverport (bezeichnet den Service)
  - Objektfeld (bezeichnet ein Objekt des Service)
  - Rechtefeld (bestimmt mögliche RPC Operationen auf Objekten oder Service)
  - Sicherheitsport der die Rechte einwegverschlüsselt enthält

Capabilities

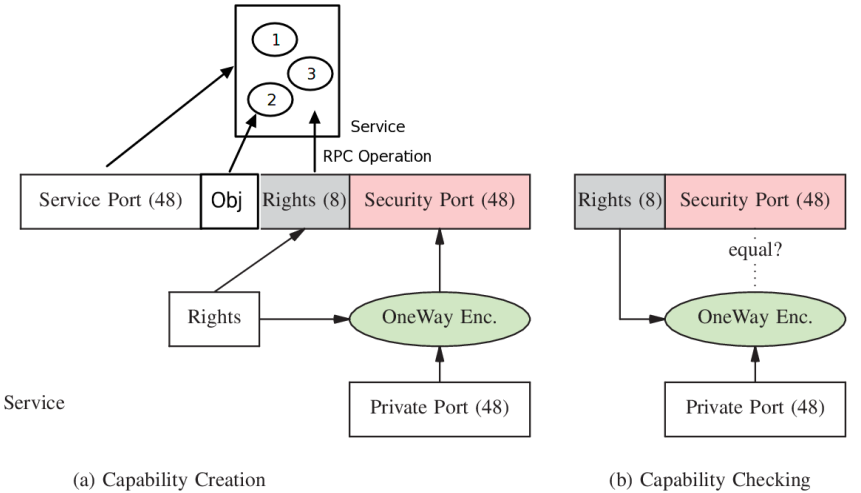
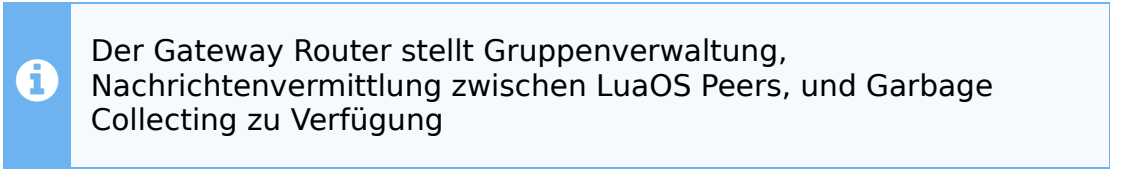


Abb. 2. Capabilities und Einwegverschlüssung des Rechtefelds mit einem (privaten) randomsierten Port

GateWay Router



- Es wird HTTP und WebSocket Kommunikation verwendet
  - HTTP für Router RPCs
  - WebSockets für Peer-to-Peer und Peer-to-Group Kanalkommunikation (aber auch hier RPC Semantik)
- Jeder LuaOS Peer hat mindestens einen Port der mit einer eindeutigen UUIDv4 bezeichnet wird
- Diese UUID kann in Gruppen eingetragen werden
- Gruppen können beliebige ID Kennzeichnernamen haben, müssen aber auf einem Router unterscheidbar sein

LuaOS Route API

- LuaOS Peers haben die folgende API:

Route
: Basisklasse umd Gruppen zu verwalten

Groups
: Gruppenzugriff und Gruppenadministration (über Route)

Ports
: Steuerung des Gateway Multiplexer, d.h., Herstellen von Kanalverbindungen zwischen Peers

Routingobjekt

- Jeder Zugriff auf den Router benötigt eine Route Instanz unter Angabe der Router URL (HTTP Port):

```
local <route> = route:new(url:string)
```

- Jede Route Instanz ist meiner eindeutigen Port UUID verknüpft; diese wird vom Router als Remote Endpoint ID verwendet; der Port muss beim Router angemeldet werden:

```
<route>:ping()
local <port> = <route>:connect()
<port>:peerid
```

Route Group API

- Mit einem Routingobjekt können Gruppen verwaltet werden:
  - `<route>:ask(groupid:string)`liefert alle Mitglieder einer Gruppe (Port UUID)
  - `<route>:create(groupid:string)` erzeugt eine neue leere Gruppe
  - `<route>:delete(groupid:string)` erzeugt eine neue leere Gruppe
  - `<route>:join(groupid:string)` fügt lokalen Peer Port zur Gruppe hinzu
  - `<route>:unjoin(groupid:string)` entfernt lokalen Peer Port zur Gruppe

Port API

- Wenn ein Peer Port beim Route registriert ist können beliebige andere Peer Ports von anderen LuaOS Knoten verbunden werden (Virtual Circuit):
  - `<port>:connect(peerid:string)`
- Wenn mehr als ein Remote Peer Port hinzugefügt wird erhält man Nachrichtenmulticasting, d.h. jede eingehende Nachricht wird an alle hinzugefügten Zielports dupliziert gesendet
- Nachrichten können über den Port empfangen und gesendet werden:
  - `<port>:write(data)` sendet eine Nachricht (String oder Datentabelle) über den vrituellen Kanal
  - `<port>:read()` liest Daten aus einer Eingabequeue die von anderen Peer Ports an diese gesendet wurden

Tupelräume

Werden noch im Modul Gruppenkommunikation behandelt

- Kommunikation von Sensorknoten über Tupelräume ist eine **Koordinationssprache**.

Es gibt Produzenten und Kosumenten. Datenaustausch ist anonym (nicht addressiert). Verbindung von Produzenten und Konsumenten über Tupeldaten nur durch deren Inhalt, nicht aber Ort.

- Die Daten sind mit Tupeln organisiert.
- Tupel-Räume stellen ein **assoziiertes Shared-Memory-Modell** dar, wobei die gemeinsam genutzten Daten als **Objekte** mit einer Reihe von **Operationen** betrachtet werden, die den Zugriff der Datenobjekte unterstützen

- Ein Tupel ist eine lose gekoppelte Verbindung einer beliebigen Anzahl von Werten beliebiger Art /Typ/
- Ein Tupel ist ein Wert und sobald es in einem Tupelraum gespeichert ist, ist es persistent.
- Tupelwerte erfordern einen **Mustervergleich** basierend auf dem *Vorlagenmuster*, bestehend aus tatsächlichen (v, ε, x) und formalen Parametern (x?, Variablen, die mit Referenzsemantik verwendet werden)
- Ein Tupel kann nur durch seine Verknüpfung mit Vorlagenmustern (Templates) p angesprochen werden.
- Ein bekanntes Tupelraum-Organisations- und Koordinationsparadigma ist **Linda** [GEL85].

Basisoperation: Output, Input, Read. Die lesenden (konsumierenden) Operation blockieren den Aufrufer solange bis passende Tupel gefunden wurden.

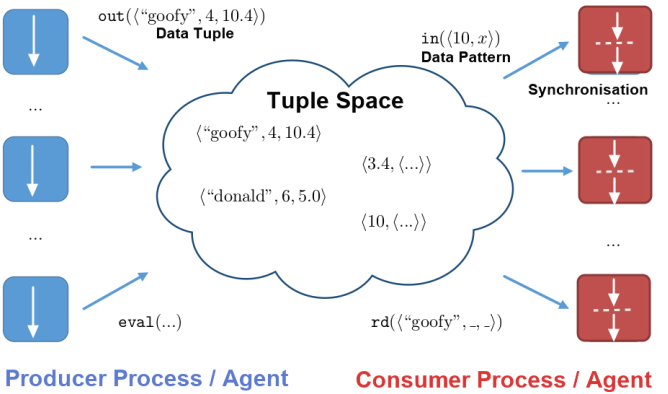


Abb. 3. Ein- und Ausgabeoperationen in Tupelräumen

Zusammenfassung

- Zwei verschiedene Methoden für die semantisch getriebene organisation der Kommunikation in Sensornetzwerken wurden vorgestellt:

1. Verzeichnisdienst mit Services über Pfade und addressierter Remote-Procedure Call Kommunikation (servicegetriebene Kommunikation)
    - Ontologien werden benötigt für ein gemeinsames Verständnis von Pfadenstrukturen, Namen, und Services
2. Datenbank mit Tupeln und rein datengetriebene Kommunikation mittels Mustern (generative Kommunikation)
    - Ontologien werden benötigt für das gemeinsame Verständnis der Daten und Suchmustern