A Lightweight Service Registry for Unstable Ad-Hoc Networks

Paulo Ricca[1], Kostas Stathis[1], and Nick Peach[2]

[1] Royal Holloway, University of London, UK
[2] PB Partnership, UK
{paulo.ricca,kostas.stathis}@cs.rhul.ac.uk,
nick.peach@pbpartnership.com

Abstract. We present a distributed systems framework for sharing knowledge and capabilities in ad-hoc networks of devices where network bandwidth, network connectivity and device computing power are severely limited. We develop a distributed registry to store knowledge of device capabilities and their invocation, implement it and show how it can be deployed in a set of network nodes to exemplify its usefulness. The ideas are exemplified with an ambient intelligence scenario known as autonomous road trains.

Keywords: distributed service registry, unstable ad-hoc networks.

1 Introduction

Ambient Intelligence (AmI) is a vision of the future where people interact with networks of computing devices, often in the form of everyday objects within a physical environment, to better carry out their everyday activities [1]. According to how objects or their capabilities are used, networks of devices enable user applications that may acquire environment knowledge via sensors, intelligently process this acquired knowledge and share it with other devices, and possibly change the environment's state using actuators.

In many AmI applications persistent connectivity, software homogeneity and unlimited computational power of devices cannot be taken for granted. As a result, how to share knowledge and capabilities between devices within an application is an important consideration. More specifically, a centralized approach is less tolerant and averse to scaling [3], while a simple custom and ad-hoc solution is not always reusable in similar applications, especially when application devices are heterogenous and their connectivity is both unreliable and dynamically formed.

We develop a distributed registry to store knowledge of device capabilities and their invocation, implement it and show how it can be deployed in a set of network nodes to exemplify its usefulness. Our contribution lies in the integration of selective distributed systems technologies combined with peer-to-peer techniques and a service-oriented approach targeted to low-powered devices. We allow nodes to register themselves or others to receive notifications when data that is applied to certain filters is created, modified or removed, inside the registry (making it easy to construct simple reactive applications as well). In addition, we support data independently of its description languages to better accommodate heterogeneous systems. The end result mixes the usefulness of a directory service, the flexibility of a distributed database and the ease of use of a data-driven query and storage mechanism, into a very lightweight peer-to-peer platform which is suitable for AmI environments.

2 Scenario

As shown in Fig. 1, we consider the operation of an Autonomous Road Train (ART), a changing set of vehicles driving in a platoon formation under autonomous control in order to reduce fuel consumption, improve safety and increase driver convenience in motorways (for more information on ARTs see [2]). When a vehicle enters a motorway where ART's are allowed, it should try to discover ARTs (step 1) through SR's and query them (step 2) in order to find out which of them would be the most useful for the vehicle's trip e.g. in terms of direction and speed. The queried SRs return data related to the state of an ART The state schema describes data relative to an ART, such as location, speed, direction, train size and position on the train sequence (slot). After choosing an ART to join, the vehicle's decision mechanism registers the synchronization and triggering mechanism (step 3) for the piece of data received by the ART (making it part of the data replication and triggering loop). Next, the car's decision module fills the ART data inserting a reference to itself on an empty slot and changing this slot's state (step 4). The synchronization mechanism ensures that all the cars in the ART are informed of this change. The vehicle can now position itself in the ART. When the vehicle reaches its destination, it changes its internal state to leave the ART (step 5), informing others of its intentions, so that the other vehicles can distance themselves (step 6) so that the car can exit the ART (step 7).

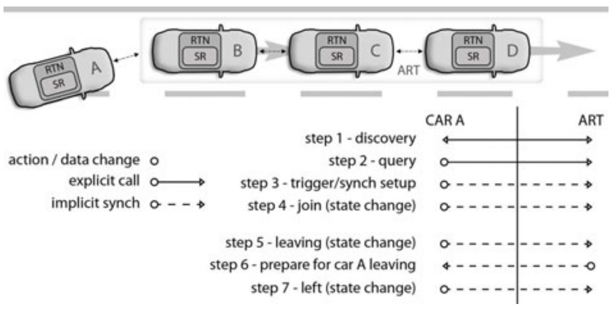
RTN
SR
A
B
C
D
ART
CAR A
ART
step 1 - discovery
step 2 - query
step 3 - trigger/synch setup
step 4 - join (state change)
step 5 - leaving (state change)
step 6 - prepare for car A leaving
step 7 - left (state change)
action / data change
explicit call
implicit synch

Fig. 1. Cars as Road Train Nodes (RTNs) with Service Registries (SRs)

3 Service Registry Prototype

One of the requirements within an ART is how to coordinate heterogenous nodes that are dynamically added to, removed from or moved about the network. Traditional co-ordination mechanisms are often achieved using a central controller that has been made robust against failure. However, using a remote radio data link to a distant central point is inherently dangerous, due to data link dropouts, therefore local connections within the ART are essential with peer-to-peer computing principles replacing the central controller system.

The Service Registry that we propose is a module of a larger end-to-end system prototype which aims at creating a highly decentralized, structured and flexible approach to orchestrate behaviour in the form of workflows. The objective is to create a visual designer tool and a run-time node to connect and coordinate low-powered and heterogenous devices on unreliable networks similar to the one discussed in [5]. In this context, the Service Registry is designed to be a custom lightweight and modular directory service structure composed of four main components depicted in Fig. 2. The Interface allows applications/devices to interact with the Service Registry. The Registry stores, searches and modifies schemas and data entities. The Query Interpreter processes requests (described on one particular protocol and data format) communicated via the Interface and interacts with the Registry accordingly. The Trigger Manager registers the interest of external components in registry events related to specific data, and notifies them accordingly when these take place. The Synchronizer performs synchronization between different service registries and, finally, the External Interface represents the Interface module of a neighbour registry.

To develop the registry we have chosen Rest [8] as a lightweight mechanism for remote and embedded local communication, Json [7] for data type and schema representation. Data is stored in a database-driven registry supported by Apache Derby, a lightweight database management system based storing and querying module. Each schema creates one main database table and one secondary table for each complex (objects or arrays) schema field, recursively for there may be other complex fields inside these, when it is registered. Below is an example of schema, Json data, internal database data and triggers used in the ART scenario.

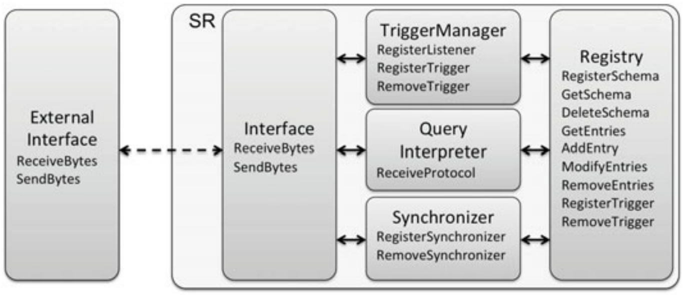

Fig. 2. Overview Implementation Architecture of the Service Registry with the description of each module's basic methods

ARTState Json schema:

```
{type:"object", properties:{slot:{type:"number"}, carId:{type:"string"},
     state:{type:"string"}, acks:{type:"array", items: {type:"string"}}}}
```

ARTState Json data:

```
{slot:0, carId:"Car A", state:"Occupied", acks:["Car A", "Car B"],
slot:1, carId:"Car B, state:"Occupied", acks:["Car B", "Car A"]}
```

ARTState internal data:

```
database table ARTSTATE:
ID; SLOT; CARID; STATE; ACKS
32; 0; "Car A"; "Occupied", 51
33; 1; "Car B"; "Occupied", 52

database table COMPLEX_ARTSTATE_ACKS:
ID;  ACKS
51; "Car A"
51; "Car B"
52; "Car B"
52; "Car A"
```

Trigger:

```
ID; GRABBERID; SCHEMA; EVENTTYPE; FILTERFIELD; FILTERVALUE
80; "Car A"; "ARTState"; MODIFY; STATE; "Occupied"
```

We use an implementation of the observer pattern to implement triggers[1]. For the above example, when the ARTState schema is registered, the registry stores the schema and creates two tables: ARTSTATE and COMPLEX_ARTSTATE_ACKS as the acknowledgements represented in a complex array. The Registry checks the data against the schema and adds the appropriate fields to the two tables created before. The trigger described above states that the entity identified by "Car A" should be notified when data related to the "ARTState" schema is modified, and the "STATE" field contains the string "Occupied". In this way, triggers allow the decision-making module of nodes to be notified of changes on the network. In our ART scenario triggers allow a car letting other cars know of its intentions to leave the ART, so that they can act accordingly, leaving enough space for it to leave securely.

4 Conclusions, Related and Future Work

We have presented a distributed systems framework for sharing knowledge and capabilities in ad-hoc networks of devices where network bandwidth, network connectivity and device computing power are severely limited. We have developed a distributed registry to store knowledge of device capabilities and their invocation, implement it and show how it can be deployed in a set of network nodes to exemplify its usefulness. We believe that such a registry is a sweet spot

[1] http://www.research.ibm.com/designpatterns/example.htm

for AmI, Internet of Things and any heterogenous distributed systems which may lack a stable network connection.

Directory Services offer a good way of storing, querying and sharing bits of structured information but they lack the lightweightness that is needed for Ambient Intelligence environments as in our case study. Lightweight Database Management Systems such as SQLLite, HyperSQL or Apache Derby provide the previously referred lightweightness required but used alone, they lack some useful features such as device notification of data update, smart synchronization between devices, data-format independence and allowing clients to use their own data formats on the registry. Data-Driven querying solutions such as OrientX [4], XPath or JsonPath allow clients to keep using their current data-formats but lack all the flexibility of directory services or common database management systems. The ad-UDDI [6] project suggest an active and distributed service registry which optimizes and extends the usage of UDDI mainly for service discovery. Although this solution offers an interesting and proven approach for active service discovery, it does not cover the issue of working on unstable networks, and as it's primarily focused on service description, it does not offer a generic solution for sharing information in such scenarios.

As part of our future work we plan to complete the synchronization mechanism provided, introduce a security layer, provide a global (ldap[2]-like) syntax for interacting with the registry and offer an accurate method for measuring results and comparing to other solutions.

References

1. Aarts, E., Harwig, R., Schuurmans, M.: Ambient Intelligence. In: The Invisible Future: The Seamless Integration of Technology into Everyday Life. McGraw-Hill Professional (2001)
2. Bergenhem, C., Huang, Q., Benmimoun, A., Robinson, T.: Challenges of Platooning on Public Motorways. In: 17th World Congress on Intelligent Transport Systems, Busan, Korea (2010)
3. Cai, M., Frank, M.: A Scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network. In: WWW 2004 (2004)
4. Meng, X., et al.: OrientX: A Schema-based Native XML Database System. In: Proceedings of the VLDB, pp. 1057–1060 (2003)
5. Peach, N.: Decentralized operating procedures for orchestrating data and behavior across distributed military systems and assets. In: Interoperability II. SPIE 8047, 80470B, Orlando (2011)
6. Du, Z., Huai, J., Liu, Y.: Ad-UDDI: An Active and Distributed Service Registry. In: Bussler, C., Shan, M.-C. (eds.) TES 2005. LNCS, vol. 3811, pp. 58–71. Springer, Heidelberg (2006)
7. Crockford, D.: The application/json Media Type for JavaScript Object Notation (JSON). Internet informational RFC 4627 (2006)
8. Fielding, R.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine, Irvine, California (2000)

[2] http://www.openldap.org/