Prototype d'architecture de découverte de services avancée

Tom Leclerc, Laurent Ciarletta, Laurent Reynaud, André Schaff

Project SARAH (Services Avancés pour Réseaux Ad Hoc)
Advanced Services for Ad hoc Networks

Working Package 3 (WP 3): Découverte de services et supervision

Partners:
Inria Madynes
Orange Labs

Prototype d'architecture de découverte de services avancée

Authors:
Chapter 1 : L. Ciarletta, T. Leclerc, A. Schaff, Madynes
Chapter 2 : L. Ciarletta, T. Leclerc, Madynes
Chapter 3 : L. Ciarletta, T. Leclerc, Madynes
Chapter 4 : L. Ciarletta, T. Leclerc, Madynes
Chapter 5 : L. Ciarletta, T. Leclerc, L. Reynaud, Madynes

Date : *03/02/2010*

Responsible of the WP 3:

Laurent Ciarletta
Laurent.ciarletta@loria.fr

INPL- INRIA Madynes

Abstract :

This document describes the implementation and tools used for the service discovery architecture described in [1]. We describe the tool, JANE, used for our simulations and the final architecture. As service discovery protocol we used Zeroconf and depict which changes had to be made to an existing implementation. We used an underlying clustering structure, NLWCA, as dissemination overlay. Finally we show the building blocks of the service discovery architechture of a device.

OUTLINE

1. General Overview

1.1. Objective of the document

(Explaining the motivation of the subject treated in the document.)
This document describes the implementation and tools used for the service discovery architecture described in [1].

1.2. Context

The SARAH project is focusing on advanced services for ad-hoc networks. More specifically, the work-package 3 (SP3) emphasizes on Service Discovery Protocols (SDPs) and architecture for those dynamic networks.
Due to the evolution of the overall project, this architecture for Service Discovery has been developed in order to be able to function with services that are being described or developed in other work-packages : SP2 (routing / multicast, geo-location) and SP4 (Security), but to not rely on such services. Its main goal is to be functionally integrated with the SP5, i.e. the demonstrator. Therefore our solution has been tailored to fit the SP5 scenarios (firstly the "Musée des Telecoms")- and the overall project but remains generic-enough to provide a solution for other settings.

1.3. References

(Documents that served as references while writing the deliverable.)

	Reference	Date
[1]	Ciarletta & al. "Architecture pour la découverte de services avancée", livrable L3.02 du projet SARAH	*2009*
[2]	D. Gorgen, H. Frey, and C. Hiedels, "Jane-the java ad hoc network development environment," *Simulation Symposium, 2007. ANSS '07. 40th Annual*, pp. 163-176, March 2007.	*2007*
[3]	Zeroconf: http://www.zeroconf.org/	
[4]	JmDNS: http://jmdns.sourceforge.net/	
[5]	A. Andronache and S. Rothkugel, "Nlwca node and link weighted clustering algorithm for backbone-assisted mobile ad hoc networks," Networking, 2008. ICN 2008. Seventh International Conference on, pp. 460-467, April 2008.	*2008*

1.4. Abbreviations / Glossary

SDP : Service Discovery Protocols. They are a sort of middleware for (software) clients to automatically discover available services (for example : printers, projectors) and for some advanced protocols to connect and use them in a transparent way. UPnP, Zeronconf, Jini are some well known SLP

SDA : Service Discovery Architecture. This is the collection of devices and services involved in the service discovery, the description of their roles, communication patterns and organization.

2. Service Discovery Architecture - Overview

The service discovery archtiecture works in pure adhoc mode while taking advantages of more stable or even fixed nodes by autonomously building a sub-set of relatively reliable nodes and links (Figure 1). By relatively we mean that have a mobility index and a weight in the network above a computed or administratively fixed threshold (Section 5).

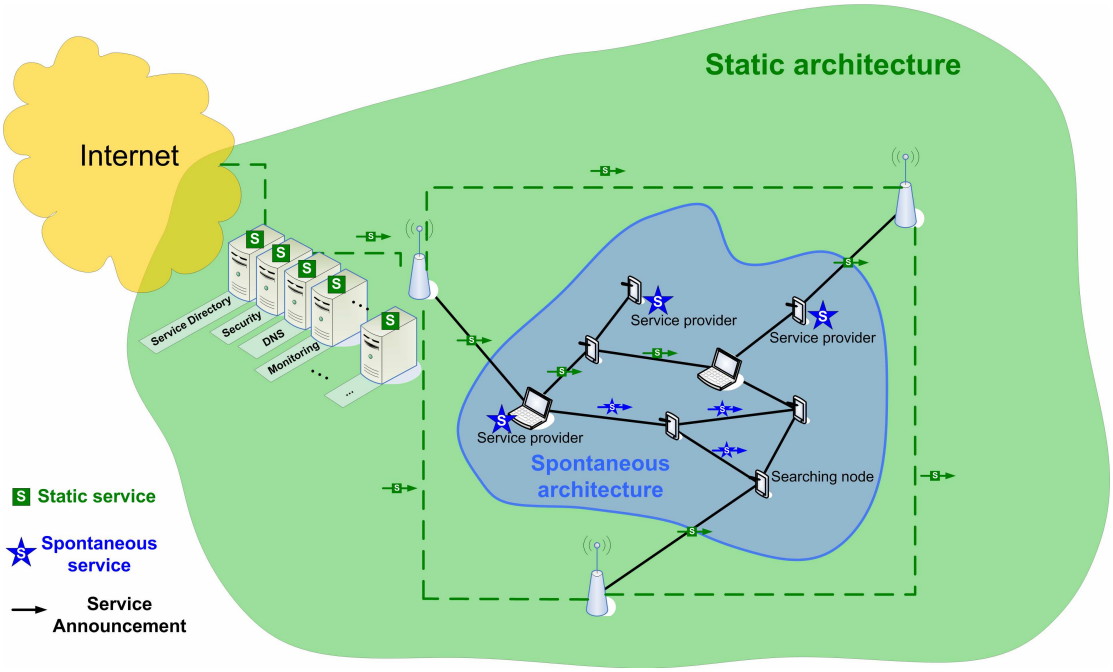

Figure 1: Global archtiecture with services

As development platform for the Service Discovery Architecture we chose the Java Ad-hoc Network Emulator (JANE).

3. Implementation tool: JANE

As development platform for the Service Discovery Architecture we chose the Java Ad-hoc Network Emulator (JANE) [2]. JANE has the advantage to be able to run in simulation and also in platform mode where the same code is used for simulation and on real devices (Nokia N800). The ability to switch from simulation to experimentation provides a fast developpement cycle from design to test. In JANE all the elements (e.g. network protocols, network applications) are represented as a JANE service. JANE is composed of three layers (*Figure 2*). The first is the platform, hybrid, or simulation core which provides its versatility. The second level is composed of the JANE operating system, providing the timer system, the service manager, the service interaction, the execution manager and the simulation knowledge which is not used in platform mode. The last layer are the services present in JANE and the developed service like new network protocols.

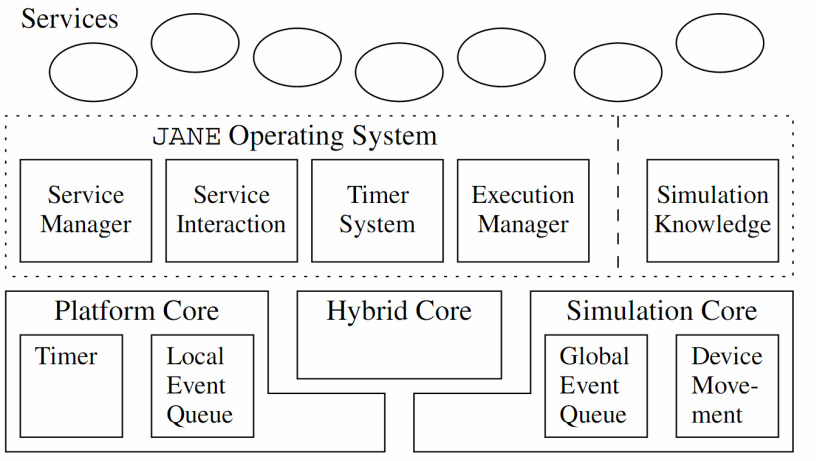

Figure 2: JANE operating system archtiecture

4. Service Discovery Protocol: Zeroconf

Zeroconf [3], short term for Zeroconfiguration Networking, is a combination of different techniques that result in a fully usable IP network without any server configuration. Zeroconf has three core elements: the automatic assignement of an IP address, the DNS functionalities (e.g. name resolution) without any central server (mDNS) and the automatic discovery of services available in the network (DNS-SD). We used the exisiting JmDNS [4] java implementation of mDNS/DNS-SD. The implementation does not contain the auto-IP part of Zeroconf.

4.1. JmDNS Code Modifications

To be able to run the JmDNS implementation of Zeroconf on JANE we had to adapt key code part: the network in/out points and the timers.

Network in/out points:
We identified every network incoming and outgoing points (e.g. sending or receiving of datagramm packets). For each points we replaced the classic java network socket with the network interface provided by JANE. Every packet, which would flow in the network through the java socket, will go through JANE. In simulation mode, packets use the simulation enviroment to flow from one simulating node to the other. In platform mode (on real devices), the change from original JmDNS implementation is minor: the packet will flow through JANE which also uses standard java sockets to send packetsto the network.

Timers:
Another important change that was made to the JmDNS code was the adjustment of all timers. JmDNS uses the standard machine time to manage its events and timers. Every code parts using a timer was replaced to use the JANE timer instead of the local machine time. In simulation mode, the time used as reference is the simulation time. In platform mode, the JANE timer reflects the machine time, thus local machine (device) time.

4.2. Zeroconf on top of a cluster structure

Zeroconf uses multicast addresses to disseminate its messages. In wired networks this is fine, multicast is widely used and deployed. In ad hoc networks it is different, the multicast structure causes more network overhead and problems due to the dynamic nature of the network. Keeping a multicast structure up to date in an ad hoc networks requires strong protocols and numerous message exchange. Therefore we replace the underlying multicast structure with a lightweight cluster structure provided by NLWCA [Section 5].

5. Cluster Structure: NLWCA

To provide a stable structure and also to provide mDNS with an alternative to multicast we use as underlying structure a clustering algorithm NLWCA (Node and Link Weighted Clustering Algorithm) [5]. NLWCA organizes ad hoc networks in one-hop clusters (*Figure 3*) by using only information available locally. Each device elects exactly one device as its clusterhead, i.e. the neighbor with the highest weight. So far, a topological chain can be formed by so called sub-head nodes. A sub-head is a node that elects a neighbor node as clusterhead but at the same time is elected as clusterhead by some other one-hop neighbor nodes. However, sub-heads can lead to more than three hops between a source clusterhead and its nearby clusterheads which could lead to a complex communication protocol. To obtain strict one-hop clusters, thus simplifying the protocol, a rule was added to the NLWCA algorithm: a node that already elected a foreign node as clusterhead is not eligible to be elected by another node as clusterhead.

The main goal of NLWCA is to avoid superfluous re-organization of the clusters, particularly when clusters cross each other. To achieve this, NLWCA assigns weights to the links between the own node and the network neighbor nodes. This weight is used to keep track of the connection stability to the one-hop network neighbors. When a link weight reaches a given stability threshold it is considered stable and the device is called stable neighbor device. The clusterhead is elected only from the set of stable neighbors which avoids the re-organization of the topology when two clusters are crossing for a short period of time.

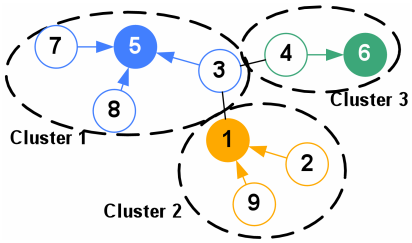

Figure 3: One-hop clusters formed by NLWCA

6. Device Archictecture

The device architechture (*Figure 4*) contains following elements:

- **Network:** underlying ad hoc network
- **Dissemination structure / NLWCA:** Logical structure which provides efficient dissemination of messages in the network. Goal: Avoid straight flooding by structuring the network and guiding dissemination of messages through key nodes (well positioned).
- **SDP / Zeroconf:** Service Discovery protocol advertising or requesting services in the network. Takes advantage of the underlying dissemination structure.
- **Filtering:** Filter certain SDP messages depending on the context to reduce computation overload. For example slave nodes only process SDP messages “recommended” by their clusterhead.
- **Context Grabber:** Obtains the context information from the SDP.
- **Context Advertiser:** Inserts/advertises context inside the SPD messages.
- **Context Manager:** Contains the acutal context compuation and analysis. Handles which context information should be advertised, which context information is important to store and which context information should be given the application.
- **Cache / Directory:** Service and Context Information storage policy system. Decides how long, how much, how detailed should information being kept. Decision is then passed to the context manager.
- **Context Agent:** Retrives and Displays application or user context information.
- **Application:** application that uses SDP and benefits from context information.

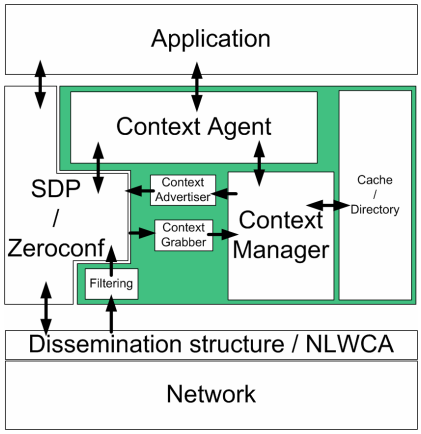

Figure 4: Device architecture

7. Prototype

As first experimentations additional to the simulations we tested Zeroconf on top of NLWCA on real devices. The devices are Nokia N800. Java is not supported officially on the Nokia’s, but the Jalimo project (https://wiki.evolvis.org/jalimo/index.php/Hauptseite) offers a free Java stack for mobile Linux devices. So it is able to run Java SE 5 applications. A good article for starting with java on Nokia N8x0 is http://www.drdobbs.com/java/208801979. *Figure 5* shows an experimentation with 3 nokia N800 running Zeroconf on top of NLWCA. In this example, Bob provides an http service. The screenshot of Alice shows the DNS table of the network, discovered using Zeroconf. Additional to the devices names, Alice’s DNS table contains the http service advertised by Bob.

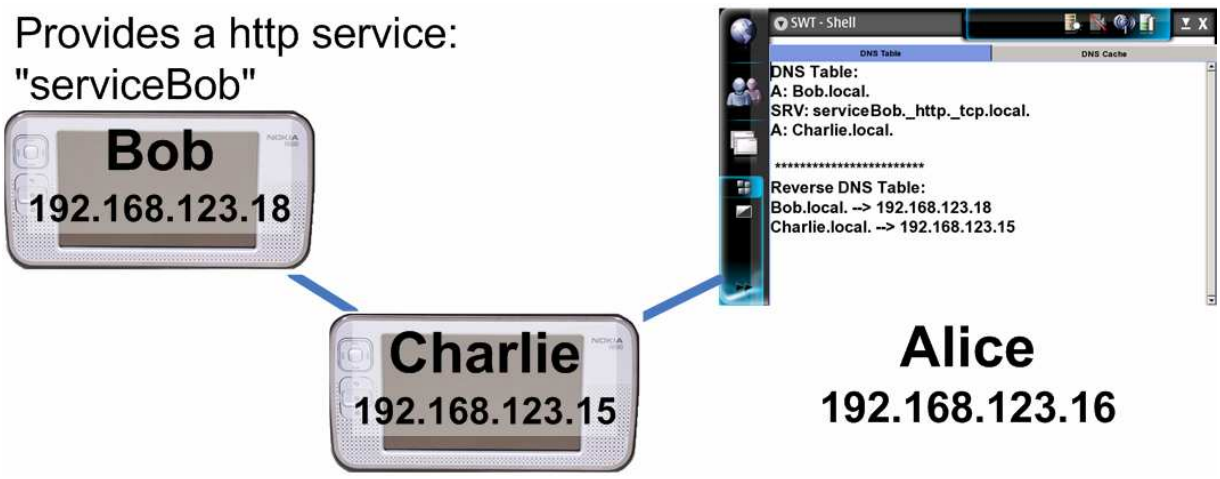

Figure 5: Experimentation with 3 nokia N800 running Zeroconf.

8. Document Management

Any remarks concerning this document should be sent to the responsible of this document.

Name: Laurent Ciarletta
Affiliation: INRIA - Madynes
Email: Laurent.Ciarletta@loria.fr
Tel: 03 83 59 20 11