# Architecture pour la découverte de services avancée

Laurent Ciarletta, Tom Leclerc, Laurent Reynaud

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

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

# Architecture pour la 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, Orange Labs*

Date : *15/06/2009*


*Partners:*
*Inria Madynes*
*Orange Labs*

Responsible of the WP 3:

*Laurent Ciarletta*
*Laurent.ciarletta@loria.fr*

*INPL- INRIA Madynes*



**Abstract** :

This document describes the Service Discovery architecture that we are proposing in the work-package SP3. Service Discovery Protocols provide a way 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. While being strongly connected to the other SPs/work-packages and most notably the SP1 and SP5, i.e. the demonstrators, it is designed to work in other settings. The foundations of this solution are its adaptability and its independence from the other services provided in the network in general, by any node (fixed/mobile or wired/wireless) or underlying layer (multicast routing, security): when they're available it can take advantage of them, but it is designed to function by building up and maintaining a relatively stable structure for Service Discovery where adhoc-specific but also standard Service Discovery Protocols and context information can be used by clients to find relevant services.

## OUTLINE

# 1. General Overview

## 1.1. Objective of the document

*(Explaining the motivation of the subject treated in the document.)*
This document proposes a service discovery architecture that supports context-aware service discovery in a hybrid scenario where spontaneously connected mobile ad-hoc nodes can take advantage of a fixed infrastructure and still being able to work without it when it is not available. In order to adapt to the availability and to the dynamicity of the network and services, we use metrics that provide contextual information that allows selecting, among the matching services, the best service in the network.

## 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. “Etude des protocoles de découverte de service dans le cadre des réseaux adhoc”, livrable L3.01 du projet SARAH | *2008* |
| [2] | Dey [2001] A. K. Dey, “Understanding and using context,” Personal Ubiquitous Comput., vol. 5, no. 1, pp. 4-7, 2001 | 2001 |
| [3] | Moustafa & al. “ Description des scenarii d'usages visés par le projet ”, livrable L1.01 du projet SARAH | 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. Related Work and foundation

In order to provide the ad hoc network with useful, user-friendly and interesting features service discovery should be provided. Service discovery facilitates resource/data/multimedia sharing or for example ad hoc/situated games, furthermore it permits to take full advantage of the dynamic networks specificities. In addition to classic service discovery, metrics that describes a nodes context permit to reduce the service selection, providing only the best service(s), according to the context in an automated way.

## 2.1. Service discovery

The goal of service discovery is mainly to find services provided by other nodes in the network in an automated way and use them only by knowing a basic set of information. Initially, service discovery protocols were designed for wired networks and most services were simple services, like for instance printing services. Not every node can or wants to achieve a given service. For example to print, a node doesn't need to be connected directly to the printer, just by using the service provided by the node that is actually connected to the printer is enough to be able to print. In the last years, a wide range of services became popular, like music sharing, game services or gateway services providing Internet access. Without infrastructure, as in ad hoc networks, the need to automatically, hence not manually (which would be too complicated and time-consuming with regards to the frequent changes in the topology), discover services that the network offers is even more crucial than in classical wired networks as no central information is available. Service discovery is even more indispensable for nodes with limited capabilities, which want to use a service without having the capability to host or run it by themselves.

Service discovery can be categorized by two items, their discovery mechanism and the presence of a service directory.

There are two discovery mechanisms, passive and active:

- Passive: the node searching a service (Client) listens passively to service announces on the network and caches them over time. When a client searches a service it consults its local cache and communicates with the matching service. The advantage of passively discovering service is only nodes providing services need to make announces on the network, thus keeping the communication low. However, passive discovery can miss idle services (i.e. the node proposing the service is not active).
- Active: the clients send a service request through the network and receive matching response in a response message. The advantage is that active discovery finds all currently available in the network matching services. However, active discovery is often too invasive.

Service directories are nodes that store, forward and respond to incoming queries from other nodes. They play the role of intermediary and ease the load of neighboring nodes. When a node sends a service query in the network the directory node instead of forwarding the query can, if it knows a matching service, respond directly to the source node. For efficiency means service directories are to be placed on relatively stable and well positioned nodes.

On Figure 1, four scenarios according to the two classifications are depicted.

- Passive discovery without directory: The service provider announces its service through the network while clients listen.
- Active discovery without directory: The client sends out a service request while matching service provider respond to it with a service response.
- Passive discovery with directory: The service provider registers its services to the service directory which then announces all known services. Client stays passive and caches the incoming service list.
- Active directory with directory: The service provider registers its services to the service directory. The client sends its service request to the directory which responds to it accordingly.

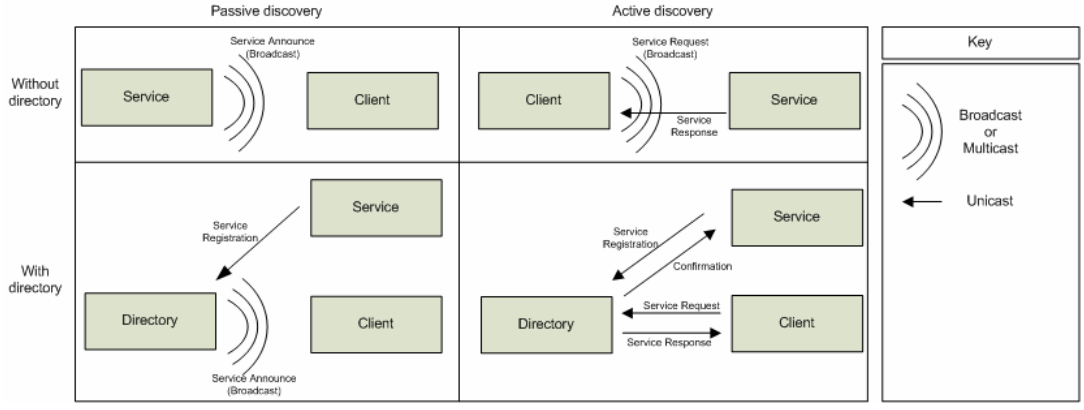


*Figure 1: Service discovery classification*

Many SDPs are not clearly reduced in one of those scenarios because of their optional use of directories or the hybrid ability for discovering a service. We considered one of those unclassifiable SDPs for our architecture because their flexibility is higher.

## 2.2. Requirements for the SDA

Our SDA has been developed using the following documents and foundations:

- Service Discovery Protocols and their application with adhoc networks are described and analyzed in former deliverable [1]. Generic SDPs are not well fitted to our requirements as they are, and other adhoc specific SDPs are not mature enough to be considered in our project.
- The SDA must be integrated and implemented with at least one of the scenarios described in [3].
- It should work well with the other work-packages contributions, using them when available but not been too much dependant on them.
- Finding the best services in adhoc networks means generally finding good enough (usable and available) services according to the node and overall context: standard performance criteria as well as context criteria (availability, proximity etc…) have to be balanced to optimize the contextual service discovery with regards to the cost of this discovery and the maintenance of the SDA
- Sometimes, providing an SDA means keeping a minimum usable service discovery alive, for example in pure adhoc setting, having a 1-hop service discovery

## 2.3. Metrics for contextual service discovery and ad hoc networks

In order to enhance our architecture with the capability for the nodes to discover matching services (matching a correct value expressing the needs, the capabilities and the context), we consider context and context-awareness as defined by Dey [2]:

“Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.”

and

“A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task.”

To be able to acquire/assess the context, we are using metrics. Metrics are different kind of quantifiable information gathered through calculation, sensors or user interaction that can characterize context information by themselves (i.e. geographical position) or be coupled with one or more other metrics (i.e. temperature sensor and humidity sensor to provide weather information) to assess the situation.

We classify metrics that may be used in our architecture in 4 categories, Network, Node, Space, and Service. Following are the metrics we see as most relevant.

### 2.3.1 Network

Metrics related to the network that can be computed or gathered from the network.

- ➔ **Hop count:** Number of nodes interconnections that needs to be crossed to reach a destination. Also called hop distance.
- ➔ **Throughput:** The amount of data that can be carried from one point to another in a given time period.
- ➔ **Bandwidth:** Maximum capacity of throughput on a given connection. It is the maximum value (theoretical) of possible throughput. This metric is generally used for the management of the network.
- ➔ **Latency:** Time elapsed between the moments the sending message function is called to the actual data transfer (i.e. processing time of the network card).
- ➔ **Jitter:** Deviation in time or displacement of data packets during transmission between two nodes. Mostly caused on relay nodes or repeater in the network.
- ➔ **Round Trip Time:** RTT, time for a data packet to arrive to its destination and come back to the source.
- ➔ **Expected Transmission Count:** ETX is the number of expected transmissions of a packet necessary for it to be received without error at its destination. This metric is implemented in OLSR.

- ➔ **Location:** Centrality measurements provide node localization in the network. Example of centralities are Degree (number of neighbor nodes),Proximity (hop distance to all nodes in the network), between-ness (number of time the nodes is on the shortest path of two other communicating nodes).
- ➔ **Link Stability:** Time or value (i.e. beacon count) a link between two nodes has stayed connected.
- ➔ **Lost Routed Packets:** Number of to be transferred packets that where not transferred.

### 2.3.2 Node

Metrics related to the node itself, mostly the characteristics of the used device.

- ➔ **CPU:** Processing power of the device.
- ➔ **Memory:** Memory initially available and left on the device
- ➔ **Signal Strength:** Signal strength of the radio signal.
- ➔ **Velocity:** Moving speed of the node.
- ➔ **Charging state:** Charging / On battery / No battery available. Gives an indication whether the node will move in the next time period (i.e. if charging, the node will rather not move a lot in the next time).
- ➔ **Moving Frequency:** Provides a movement profile of the user and helps on movement prediction.
- ➔ **Moving Tendency:** By coupling the charging state and the moving frequency one can predict more accurately the moving tendency of the node in the future.
- ➔ **Cardinal Direction:** Could be provided by a compass. Helps predicting the next location of the node.

### 2.3.3 Space

Metrics related to the space/environment surrounding the node.

- ➔ **Relative Mobility:** Mobility of a node relatively to other surrounding nodes. Nodes moving together should not be penalized when moving in groups.
- ➔ **Geo-localization:** Absolute or relative position of the node in space.

### 2.3.4 Service

Metrics related to the services provided on the network.

- ➔ **Service proposition:** Number of services the nodes provides to the network.
- ➔ **Service knowledge:** Number of foreign services the nodes knows.
- ➔ **Service discoverer:** Context in which a service was discovered. This metric is composed of:
  - o **Service discovery type:** Did the service discovery occur passively or actively.
  - o **Discovery time:** Time required for the discovery of the service
  - o **Service distance:** Hops needed to communicate with the service.
  - o **Service usage:** Gives usage information as, duration of usage, last time used and usage frequency.

## 2.4. Context as a metric and filter for service discovery

In our architecture, we want to use contextual information as a filter and parameter for service discovery. Context helps improve the accuracy of a “reply” to a given service discovery query: it should help find the “best service” given the context (given a set of contextual and usual metrics).
When used as a parameter, it supposes that those parameters are defined within the SDP framework.
It can also be seen as a filtering operation. In that case it can be seen from two points of view, depending on where the filtering takes place:

- Context information can be computed or infered using local or global information: service provider/directories and user/application have a local agent enhancing standard request with context parameters and filtering replies using contextual metrics

- Context is used as an intrinsic filtering mechanism: intermediate nodes acts as filters/proxies on standard queries and/or replies: they relay requests and replies, or stop their spreading according to the context.

3 levels of contexts are to be considered, when available:

- the user / requester : its context, for example where it is located, what are its capabilities.
- the service provider : for example how often and how fast it works, its actual computing/network charge
- the overall context of the network could also be considered (global network usage, stability, etc.).

Context can be defined as “functions”, tuples (key – value) or in an object oriented way:
In order to keep the solution simple, the (key-value) approach is preferred (see section 4).

# 3. Service Discovery Architecture

## 3.1. Specifications

SARAH is mainly about advanced services for adhoc networks, with a couple of applications and use cases in mind (see deliverable 5.01 [3]), and a set of constraints due to the evolution of the other work-package goals and deliverable.
This has given us a set of SARAH specific requirements that are driving our technical and scientific choices:

- The environment is dynamic and can have several different types of nodes: some are fixed, some are highly mobile, some are wirelessly connected in a multi-hop adhoc way, some are almost always on and interconnected by a stable network, and overall they may have heterogeneous capabilities (see Figure 2)
- The underlying adhoc routing protocol used by the project is OLSR, but we want our solution to work even in the case of other protocols.
- Applications should be context-aware, for a given set of context information (localization, etc.): the focus is not on complex semantic, which is out of the scope of SARAH, but rather on mechanisms that augment the representation of the environment and situation that help optimize service discovery and management with a limited set of extra information.

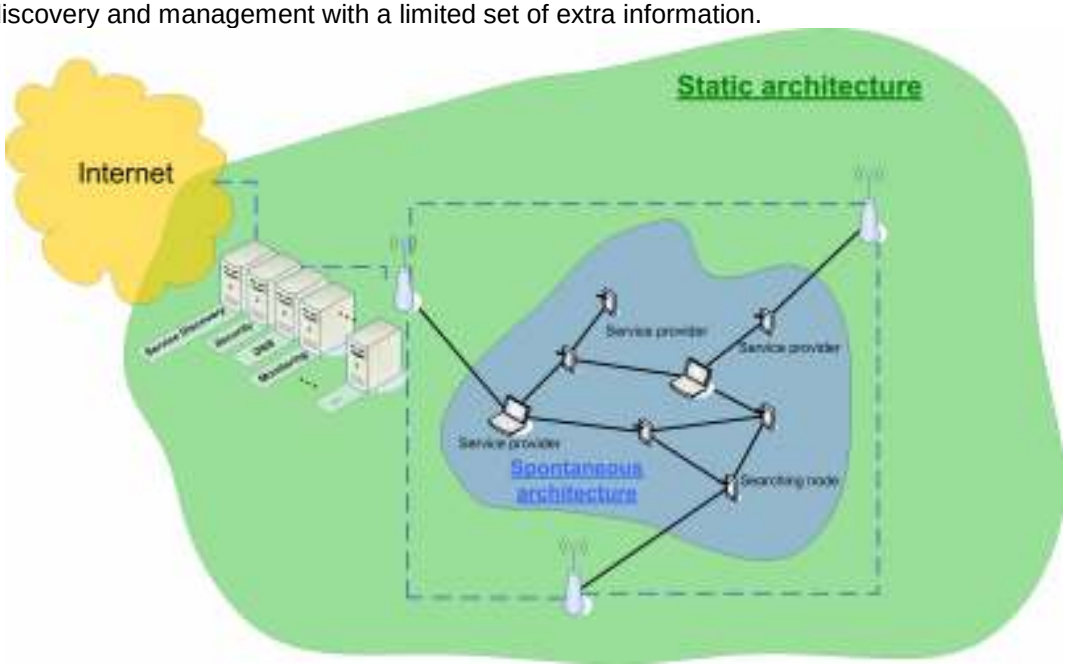


*Figure 2: Global architecture*

## 3.2. Chosen architecture

At first, we consider that when the network starts, services are available and announced (Figure 3), and users start requesting some of them, like would happen in any network. The standard service discovery protocols (UPnP, ZeroConf, Jini) can work out of the box, although they will operate within the constraints

given by the availability and performance of the services provided at the network level: (bandwidth, naming/DNS, multi -network interfaces, link local/global routing, local broadcasting or IP multicast etc.).
It is for example very important to note that if an efficient IP multicast is provided, most of the standard SDP can be used with no major modification for the entire network and will automatically take advantage of the underlying overlay.

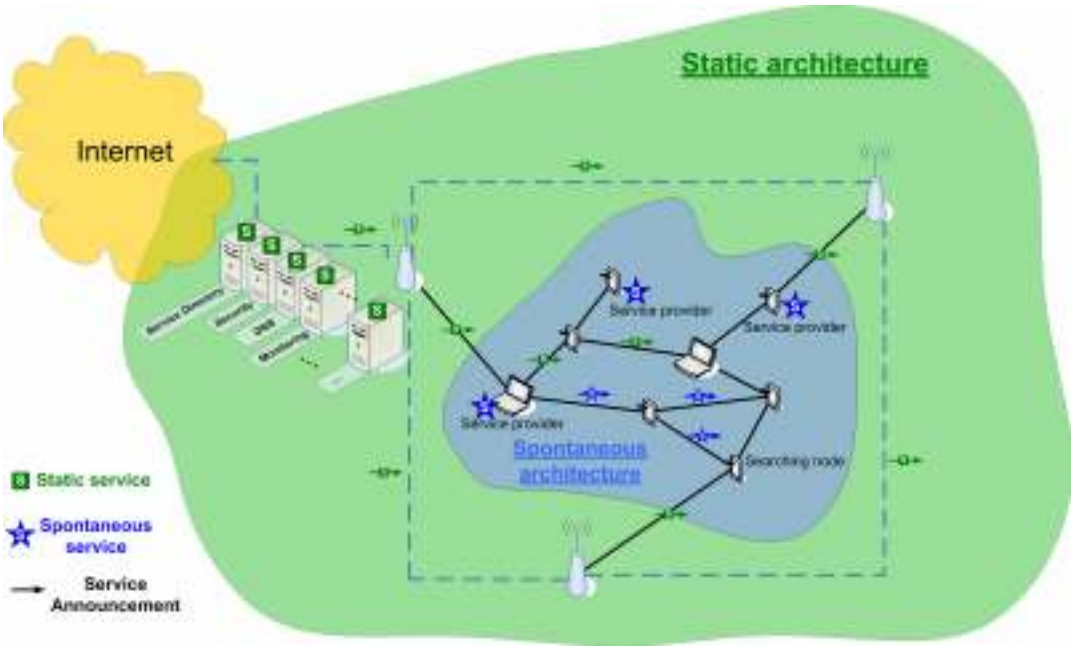


*Figure 3: Global archtiecture with services*

Our solution works in several modes: from pure adhoc (Figure 4) to Mesh network (Figure 2). It is adaptive relatively to the dynamisms of adhoc networks. It works in pure adhoc mode (Figure 4) while taking advantages of more stable or even fixed nodes by autonomously building a sub-set of relatively reliable nodes and links. By relatively we mean that have a mobility index and a weight in the network above a computed or administratively fixed threshold.

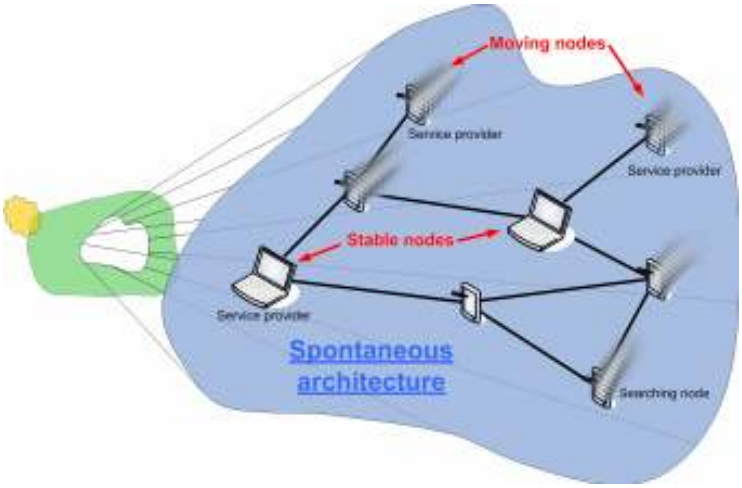


*Figure 4: Spontaneous architecture with stable and moving adhoc nodes*

In order to improve the performance (efficiency in the sense of context accuracy / flooding limitation etc) of the service discovery, when necessary and possible, we are adding Service Discovery Helpers.

This can be described as a mechanism related to “topology control”: when there exists a certain stability of parts of the network (nodes, sub-nets, global adhoc net etc…), directly measurable or inferable, some well positioned (in the network topology) and stable-enough points are chosen as “Service Discovery Helpers”.

This stability can be permanent and administratively chosen (fixed nodes that have permanent power form an outlet), or computed (nodes that have a high centrality/weight or that have been having enough resources for the longest time),
They play both a role of Service Directory and of Cache depending on the deployed mechanisms: for example if the SDP uses a Directory, in addition to administrator based directories, new ones are automatically deployed, or if the SDP doesn't use a directory, but broadcasts or multicasts its requests, some nodes acts as relays or proxies building a sort of context aware-multicast tree for the diffusion of the request and the collection of replies, and some others acts as cache that stores context related service information.

Examples :
With a Jini-like protocol.
When a node has the given "stability" and "serviceability", and if it is well positioned in the network (it has a good centrality (between-ness for example)) it will start to behave as a Directory (Lookup Service) and communicates with other serviceable nodes to form a backbone of directories maintaining the coherence of the services availability and limiting the amount of requests or announces "flooding the network".

With a DNS-SD-like protocol: these nodes will act as caches/intermediary building the needed Service Discovery layer to give accurate replies by minimizing the flooding of the network (e.g. in case some multicast mechanism is available), and exchanging with other serviceable nodes the minimal info needed to keeping a valid view of service availability. A detailed description of DNS-SD is given in section 4.
Those nodes could also play a role in the filtering / context process, by making decision about forwarding requests or relaying replies.

The architecture can be seen from different points of view.

Firstly, we can consider the following different layers (Figure 5):

- Network: this layer is providing the underlying services needed for our SDA. For example network services such as routing, multicast, dhcp, DNS etc.
- Context Agent and Service Discovery Protocol: this is our main layer, it can be adapted, optimized and enriched according to the availability of the Network services and of context inforamation, but should still work in their absence
- Application: in this layer, context-aware applications can be built using Service Discovery, providing on demand high-level services

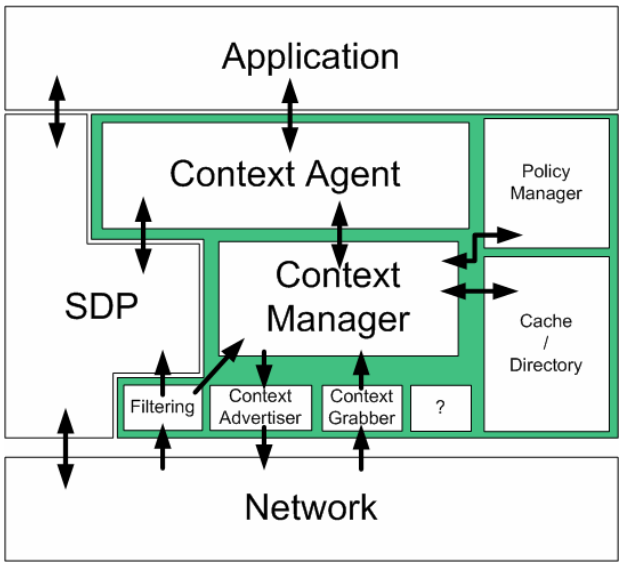


*Figure 5: Device architecture*

Secondly, we can functionally organize the architecture with "domains" (Figure 3):

- First, we have can be seen as legacy or accountable domains
  - Servers or nodes that are always on and provide standard services
  - The fixed wired and wireless infrastructure
- Second we have the spontaneous domains
  - A sub system composed of stable-connected elements
  - And dynamic elements that are moving around the former ones (example in Figure 4)

Generally speaking, global services will be provided in legacy domains:

- Global services are services that are managed by the owner of the stable architecture and they are supposed to be available most of the time.
  - DHCP / auto-configuration
  - DNS
  - Security: as defined in SP4 etc…
  - Service Discovery
  - Context Information
  - Management
  - Web
  - …
- Local services will be available as a function of the mobility, stability and context of the spontaneous domain

The Nodes are of several types:
- Service provider: global or local
- Context provider: global or local
- Client
- SD Helpers

But in the spontaneous domain, we can't expect to have special nodes playing permanlty dedicated roles

With this architecture, we'll have to develop a metric that will evaluate the network and subnetworks state at different point in time.
The network can be:
- At "startup"
- Stable
- Very dynamic
- In an emergency situation
- Etc.

# 4. Potential Service Discovery Protocol: DNS-SD

We plan to use DNS-SD for service discovery. DNS-based Service Discovery (DNS-SD) is a convention for naming and structuring DNS resource records. It is not a service discovery protocol as is but, since it is based on the DNS protocol widely implemented and used, standard DNS query are used for service registrations and queries. There are multiple advantages using DNS-SD. Existing DNS servers can easily contain service records and interact to queries and registrations. In case no server is available, DNS-SD adapts easily to ad hoc multicast mechanisms (as multicast DNS (mDNS) or another protocol).

For example a printer service: "_ipp._tcp.example.com", where ipp is the printer protocol with tcp used and finally the domain name example.com. The result of a query searching for a printer (ipp protocol) is a list with available printers in the network. To obtain a list of printers matching more specific needs (Paper size, color, etc.) the TXT record (as given in the standard DNS) contains the detailed information about a DNS record (here service records). The TXT record can contain any necessary information that can be useful for service selection. The content of the TXT record can be easily adapted with new metrics describing the context of a service. The TXT record can be obtained by sending a query or can be added to the initial response message of the service query.

We propose to use the TXT record of DNS records to incorporate context information. We define a context information as "functions", tuples (key – value). For example for context information about localization the TXT record would be added with the String: "localizationGPS.Latitude:48.8566670.Longitude.2.3509870" where localization is the key for a gps enabled device position information.

# 5. Information Management: perspectives

The amount of information available in such dynamic network can become very large, especially if we are looking at multimedia data, when considering the overall network. But only a small part of it is actually interesting regarding the nodes context (location, resources, usage). Firstly, there is a need to reduce both

the incoming/shared information at the node level and the amount of data exchanged to maintain usable information. Secondly, static management systems outside of the ad-hoc network may want to collect information that is of no interest for the nodes inside the ad-hoc network. In order to optimize the data exchanges, existing techniques may be used to filter information at the node level, by building an information path, similar to a multicast tree for example, in which information flows only to the interested nodes.
As for now, this is outside of the scope of this architecture, but it seems reasonable to explore this problem in the future.

# 6. 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

# 7. Annex

*(The annex is optional. Some parts (detailed comments, tables, figures … ) could be put in the annex for better readability of the deliverable (if needed).)*