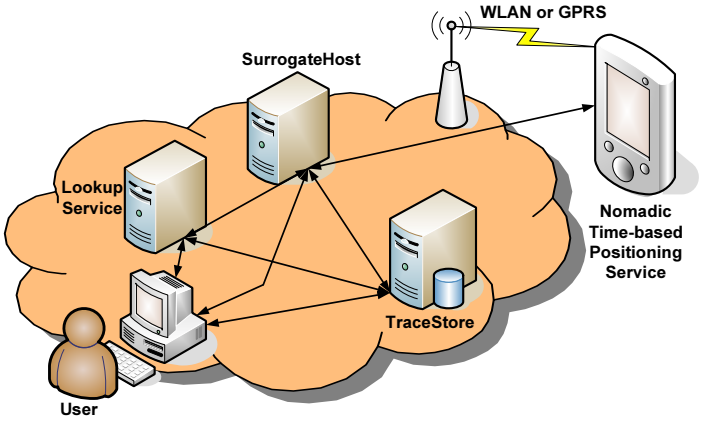
UNIVERSITÉ DE GENÈVE

University of Twente
The Netherlands

Design & Implementation of a Location and Time Specific Context Source



Integration of Time-Awareness in the Existing Nomadic Positioning Service

Laurent Kolakofsky

Contents

Preface

This thesis is the final report of the project completed as a partial fulfillment of the requirements for my License diploma in the Information Systems, oriented in Technology. My work includes the comprehensive research, design and implementation of a Nomadic Time-based Positioning Service (i.e. time-aware) based on the existing Nomadic Positioning Service (i.e. location-aware). The latter is based on a combination of Place Lab positioning software with a Jini Surrogate architecture. This thesis researches the most efficient solution to integrate time-awareness in the existing location-aware service.

This thesis was completed at the University of Twente in Enschede (the Netherlands) between the October 2005 and March 2006 in the Department of Electrical Engineering, Mathematics and Computer Science, and Architecture and Services of Network Applications research group.

I would like to thank my coordinator Kate Wac, my supervisor Aart van Halteren for their comments and help. I enjoyed working on this assignment and was able to finish it thanks to them.

Laurent Kolakofsky

Geneva, 25^{th} March 2006

Chapter 1

INTRODUCTION

Nowadays, more and more people own a personal mobile device such as a *Personal Digital Assistant* (PDA) or a Smartphone, which are equipped with wireless networking capabilities such as WLAN, 2.5G, 3G and Bluetooth. Those mobile devices make possible a new kind of mobile/ubiquitous services that are aware of their users' environment. Those are called context-aware services, and the more information a service utilizes from the user environment, the more customized and personalized at run-time to the user's needs, the service can be. There exist some location-aware services, i.e., aware of the geographical location of the user; aware of "where" they are used. These services are mainly used for tourist guidance systems in an unknown location. We argue that these location-aware services can be highly improved by adding a time-awareness; this way the service would not only know "where" it is used but also "when".

This term paper's work has been carried out as a part of the Dutch Freeband *AWARENESS (Context AWARE mobile NEtworks and ServiceS)* research project.

1.1 The Freeband Awareness Project

1.1.1 *Introduction*

Increasingly mobile devices, sensors and consumer electronics are equipped with (wireless) networking capabilities. These devices communicate via different types of networks, and together enable a complete new generation of applications: *context-aware and pro-active applications.*

Context-awareness paradigm is used by applications that have information about the circumstances (i.e. context) under which they operate and can react accordingly to this context and to the changing user needs.
Pro-activeness paradigm is used by applications that act in advance to deal with an expected event, or context change.

The context-aware applications make use of the context of their users, and of the resources (in terms of e.g. network capacity, devices) that are currently available at user's location and time. For example, a context-aware application could find the nearest restaurant according to user's food-type preferences, his current geographical position and a time of the day.

The Freeband AWARENESS Dutch national project [AWA01] focuses on service and network infrastructures that are needed to support delivery of context-aware and pro-active applications over heterogeneous mobile networks. Particular attention is given to mobile health applications for tele-monitoring and tele-treatment services [AWA02].

Eight Dutch organizations participate in the AWARENESS project: Lucent Technologies, University of Twente, Roessingh R&D, Telematica Institute, Twente Institute for Wireless and Mobile Communications, Ericsson, Yucat, Twente Medical Systems International [AWA02].

1.1.2 *Project's Goal*

The goal of the Freeband AWARENESS project is to research and design a service and network infrastructure for context-aware and pro-active mobile applications, and validate this through prototyping in a mobile health domain. In the AWARENESS project vision, a human user is always and everywhere surrounded by a networking environment ('ubiquitous') that is able to determine the identity of the user and the (upcoming) context information that is (or might become) relevant to service provisioning ('attentiveness'), such that the user can have anywhere, anytime access to mobile services in a secure and privacy-sensitive manner. One of the results of the project will be the Integrated Health Demonstrator using proof-of-concept software components [AWA02].

1.1.3 *Architecture*

The AWARENESS architecture consists of three layers: the mobile applications layer, the service infrastructure layer and the network infrastructure layer (Figure 1.1).

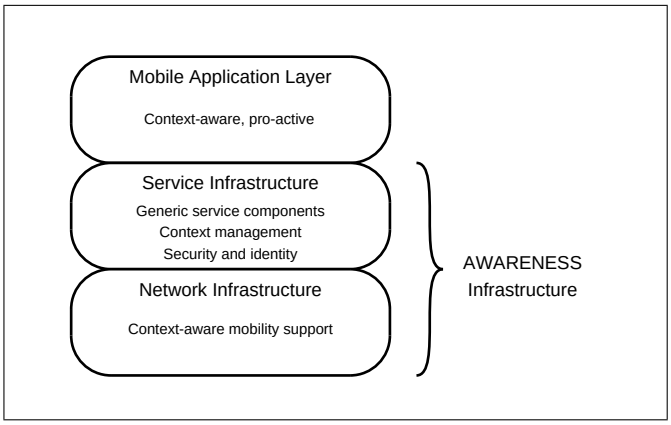


Figure 1.1: AWARENESS Infrastructure [AWA02]

Starting from the bottom, the network infrastructure layer offers seamless mobile connectivity (with use diverse wireless network technologies e.g. 2.5G, 3G or WLAN) to the service infrastructure layer.

The *service infrastructure layer* provides execution environment for mobile services. It consists of generic service components that support rapid development of ubiquitous attentive applications. These support functionalities include context management, intelligent context information processing, federated identity management, 3^{rd} party access control, mobility management, service discovery, privacy enforcement and security mechanisms.

In the AWARENESS project, the network and service infrastructures will be validated through prototyping in a mobile health domain; therefore, the *mobile application layer* will be implemented in this specific domain. The aim is that the implemented health applications will support tele-treatment of patients. Therefore, a part of the mobile health service platform is a *Body Area Network* (BAN) consisting of sensors and actuators worn by a patient collecting vital signs data and making them available in realtime to healthcare professionals in health care centres [MPM01].

1.2 Nomadic Services

1.2.1 *Overview*

The AWARENESS project focuses on service and network infrastructures that are needed to support delivery of context-aware and pro-active applications over heterogeneous mobile networks. Therefore, the project offers an innovative service infrastructure named *Mobile Service Platform* (MSP), which facilitates both: usage and provisioning of mobile services by mobile devices. In this context, nomadic services are services provided by mobile devices. The MSP simplify the development of nomadic services, and it takes into account the fact that mobile devices use wireless networks that give no reliability guarantees.

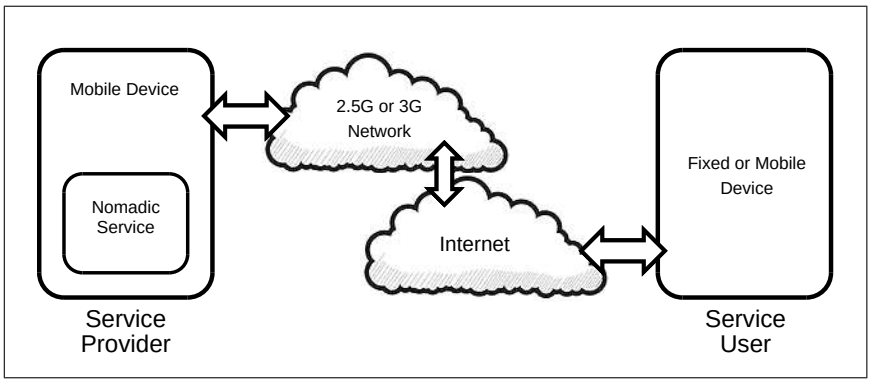


Figure 1.2: Nomadic Service Architecture [NPS01]

Figure 1.2 presents a nomadic service provider offering a service on a mobile device. The service may be offered through a 2.5G or 3G mobile operator network as well as through any other wireless network supporting user mobility. A mobile network is then connected to the internet, via which the service user on a fixed or mobile device can use the nomadic service [NPS01].

1.3 Nomadic Positioning Services

In this section we present the existing Nomadic Positioning Service (NPS) developed in frame of the preceding student's project [NPS01].

1.3.1 *Architecture Overview*

The Nomadic Positioning Service implements location-awareness and particularly it aims in determining a geographical location of a nomadic service provider. This service is composed by combination of positioning software and the Mobile Service Platform (MSP), which includes service discovery mechanisms.

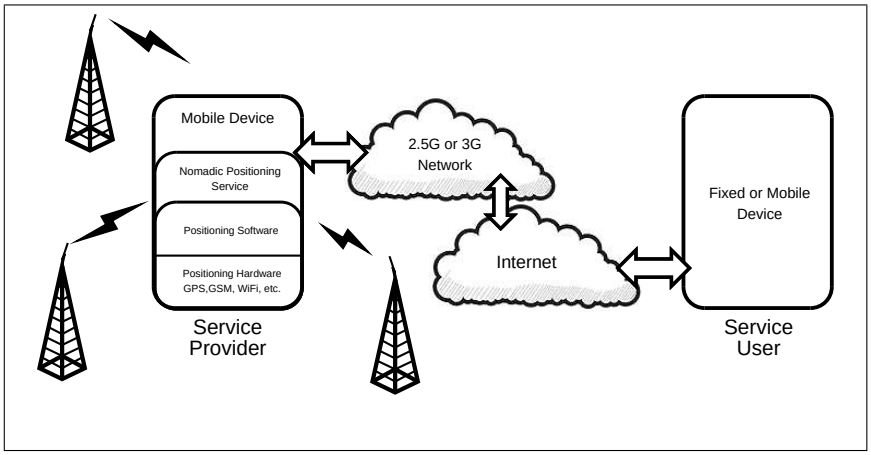


Figure 1.3: Nomadic Positioning Service [NPS01]

Figure 1.3 presents an overview of the Nomadic Positioning Service. The positioning software deployed at the mobile devices, estimates the location of the nomadic service provider by analyzing particular parameters related to the used wireless technology. These parameters include detected wireless (radio) beacons[1], their response time and signal strength. The MSP service discovery architecture allows the nomadic services to be advertised to its potential users on the (mobile or fixed) internet. This architecture shields from users the fact that in a mobile environment a nomadic service provider may change address at any time due to roaming through different mobile operator networks.

1.3.2 *Basic Features*

The Nomadic Positioning Service allows mobile devices (nomadic service providers) to provide their location to a (mobile or fixed) service user. Each

[1]Radio beacons are transmitted by a radio transmitting station and those normal radio signals can be used for determining the direction or position of those receiving them.

nomadic device provider provides its own location information. There are two modes of operation. Firstly, a service user can subscribe to a mobile device's location service such that he will be notified on every location change. In a second mode, a user can make a request for the last known position of a nomadic service provider. A service user can request location information to as many service providers as he needs. When a mobile device stops providing a nomadic service, for example due to a wireless network failure, within a timeout this nomadic service will disappear from the service listing advertised to its users.

1.3.3 *Design Features*

The main design features of the Nomadic Positioning Service are [NPS01]:

a) Efficient use of bandwidth - the amount of data exchanged between the service provider on a mobile device and the user on the (mobile or fixed) internet can be adapted to the capacity and cost of the available wireless technology.

b) Numerical scalability - the bandwidth usage on the wireless link does not increase with the number of parties interested in the location of a mobile device.

c) Plug-and-work - no additional configuration is required for publishing services when new nomadic service providers, i.e. new mobile devices come online.

For more information about Nomadic Positioning Service requirements, its features and development process please refer to the preceding student's project report titled "Nomadic Positioning Services for a Mobile Service Platform" [NPS01].

1.3.4 *Problems*

The major problem related to the current implementation of the Nomadic Positioning Service we address in this work is the fact, that this service implements only a location-awareness but not a time-awareness. That means that in the current implementation of the system, a service user can get only the location of the nomadic service provider, without the indication of time, for which this location is (or was) valid. The current implementation of the service, it does not have any kind of time management or even a reliable time

source, except from a mobile device's clock, which has no guarantee of being valid or precise. The problem we indicate is that in the current implementation of Nomadic Positioning Service, there is no reliable timestamp associated to a location estimation provided by nomadic service provider. Moreover, there is no guarantee upon clock synchronization between nomadic service provider and the service user. This can be an obstacle for some time-critical applications. For example, in the healthcare domain, where the service reliability may decide upon one's life, time-awareness is required. Imagine the tele-monitoring service used for diabetic patients, in which healthcare professional may continuously monitor sugar-level in the patient's blood and depending on it, he can remotely decide upon injection of insulin through a insulin pump directly connected to a mobile device worn by the patient. In such a case, a lack of time-awareness of events (when exactly sugar-level in the patient's blood was too high/low) and then time synchronization (how much time it takes for a healthcare professional to react) between patient's mobile device and healthcare professional device may endanger patient's life.

Another problem related to the current implementation of the Nomadic Positioning Service we address in this work is that the accuracy of location estimation can vary depending on the environment (i.e., the number of known radio beacons in the area) and the algorithm chosen for this estimation. Depending on the application, the accuracy of the position may be a very critical parameter. Unfortunately, the existing implementation of the Nomadic Positioning Service does not provide it reliably to the service user.

Another very important problem we address is that in the current implementation of Nomadic Positioning Service there is no storage of history of nomadic service provider locations; the service user gets only the location change for the ongoing session, and no historical data can be provided to him. Particularly, when a service user subscribes to a Nomadic Positioning Service, he has to always firstly request the location nomadic service provider and then wait for the service provider (a mobile device) to publish its actual location. It means that in case if the nomadic service provider is offline, i.e., has no wireless connectivity, the service user will not be able to get its last known position, because it is not stored by the service.

1.4 Nomadic Time-based Positioning Service

1.4.1 *Introduction*

To deal with highlighted problems of Nomadic Positioning Service, we propose to take into consideration time as an additional dimension of the nomadic service positioning. In this section, we present shortly our proposed solution,which we denote as the *Nomadic Time-Based Positioning Service*. Further details on this service we provide throughout the whole document.

1.4.2 *Combining Nomadic Positioning Service and Time-awareness as a solution*

To integrate time-awareness in the Nomadic Positioning Service, we must integrate it into the nomadic service provider and into the service user components.

At the nomadic service provider side, we need to timestamp every location-request event and send the timestamp together with location information to service user. In order to get reliable time-aware service we need to ensure that service user's and provider's clocks are accurate and synchronized.

At the user side, integration of time-awareness would give the user the possibility to request the location of the nomadic service provider for particular time instance, which can be defined in the past as well as in the present. This implies that the service to store location events together with their timestamps, such that they can serve later user's requests. We proposed the development of the component that stores location events and their timestamps and we denoted is as *TraceStore*.

1.5 Research Questions

Adding time-awareness into the Nomadic Positioning Service required study of its current architecture. Based on that study, we have enumerated identified problems (as indicated in section 1.3.4) and sketched proposed solution by defining some of the main features of the Nomadic Time-based Positioning Services (as indicated in section 1.4.2).
To pursue our solution further, we have identified four research question domains. Firstly, we need to focus on the nomadic service provider/user clock synchronization, secondly on the time-awareness of the nomadic ser-

vice provider component, thirdly service user's time-awareness and finally on the efficiency of Nomadic Positioning Service redesign. Therefore, we consider the following research questions.

Research questions concerning nomadic service provider/user clock synchronization:
1) *What is the most efficient way to accurately and reliably synchronize service user's and nomadic service provider's clocks?*

Research questions concerning the nomadic service provider's time-awareness:
2) *What is the most efficient way to integrate timestamp in location events?*

Research questions concerning the nomadic service user's time-awareness:
3) *What is the most efficient way to log all time-aware location events and provide them on user's request?*

Research questions concerning efficiency of Nomadic Positioning Service redesign:
4) *What's the most efficient way to redesign the existing Nomadic Positioning Service while keeping its design features such as an efficient bandwidth usage, a numerical scalable architecture and lack of need for an additional configuration when new devices come online ?*

1.6 Approach

Different approaches can be taken to answer the identified research questions. We have chosen firstly to study the design and implementation of existing Nomadic Positioning Service and its direct background components, in particular the Jini Service Oriented Architecture and Place Lab. Based on this background knowledge we propose redesign of Nomadic Positioning Service such that it integrates time-awareness at the nomadic service provider and user sides. Furthermore, we design TraceStore component that stores location events and their timestamps as provided by nomadic service providers. Based on the evaluation of the prototype developed by us, some of the possible value-added services will be indicated and the conclusion will be given in the last section of this document.

1.7 This Document's Structure

Chapter 2 presents the results of our background study on Nomadic Positioning Service and technologies required to combine it with time-awareness.

Chapter 3 presents requirements and high-level design for a Nomadic Positioning Service integrating time-awareness, which we then denote as the Nomadic Time-Based Positioning Service.

In Chapter 4, we evaluate the Nomadic Time-Based Positioning Service prototype.

In Chapter 5, we conclude on our work and provide some future research domain.

Chapter 2

BACKGROUND

2.1 Introduction

This chapter presents the results of our background study on the existing Nomadic Positioning Service and technologies required to combine it with time-awareness.

2.2 Software Platform

This section will briefly present the Java 2 Platform used for the Nomadic Positioning Services implementation. For all components, the second edition of Java 2 Platform was used, i.e., the Standard Edition (J2SE), except for the nomadic service provider component, which uses Java Micro Edition (J2ME). This exception is dictated by the fact that nomadic service provider is running on J2ME-enabled mobile device such as a PDA.

The portability of the Java Platform allows the Nomadic Positioning Services to run on most of the operating systems. The hardware on which runs the nomadic service provider must has a wireless controller supported by Place Lab's native code (see section 2.2.2 for details on Place Lab).

2.2.1 *J2ME*

Micro Edition of the Java 2 Platform (J2ME), is a collection of Java Application Protocol Interfaces (APIs) targeting embedded consumer products such as PDAs, cell phones and others mobile devices. J2ME-enabled mobile device implements a profile. The most common device's profile are a) the Mobile Information Device Profile (MIDP) aiming at mobile devices such as

cell phones, and b) the Personal Profile (PP) aimed at embedded devices like set-top boxes and PDAs. A profile is a superset of a configuration. There are currently two configurations: Connected Limited Device Configuration (CLDC) and Connected Device Configuration (CDC).

The CLDC configuration contains a strict subset of the standard Java class libraries, and is the minimal needed for Java virtual machine to operate. The CLDC is used to classify the myriad of devices into a fixed configuration.

The CDC contains a larger subset of the standard Java class libraries; it contains almost all the libraries that are not Graphical User Interface (GUI)-related.

As the MIDP is designed for cell phones, it targets an LCD-oriented GUI API. Newer, version 2.0 MIDP includes a basic two-dimensional (2D) gaming library and applications written for this profile are called MIDlets.

The Personal Profile includes a more comprehensive Abstract Window Toolkit (AWT) subset and adds applet support [J2ME01].

In our project, we use the CDC profile and the CLDC profile of IBM's J9 Java virtual machine (JVM) for the nomadic service provider, which allows the service provider to run on PDA's.

2.3 Positioning

This section presents the positioning software and hardware used by or related to the location-awareness provided by the Nomadic Positioning Services: JSR 179, Place Lab and GPS.

2.3.1 *JSR179*

The *Java Specification for Requests* (JSR) #179 is a Location API for J2ME. This specification defines a J2ME optional package that enables mobile location-based application run on resource-limited devices, e.g. mobile terminals. Particularly, this API produces information about the geographical location of the mobile terminal to Java applications by accessing a database of known landmarks. A landmark is only an information container; it stores the place's name, its description, its coordinates (and optionally its address). The minimum platform required by this JSR is the J2ME CLDC v1.1 or

J2ME CDC [JCP01]. Our Nomadic Positioning Service meets this requirement and already implements this JSR through Place Lab, which implements it. However, the landmarks are not used now in the Nomadic Positioning Service because the coordinates provided by the Place Lab contain enough location-related information and we do not need to associate them a name, a description or an address.

2.3.2 *Place Lab*

Introduction

Place Lab is Java-based software developed by Intel, which provides a mobile device positioning service for location-based mobile applications [PLA01]. Place Lab is able to determine location of the terminal (see Figure 2.1) with the aid of nearby radio sources, such as fixed Bluetooth devices, 802.11 (i.e., WLAN) access points, and GSM cell towers. Particularly, the Place Lab software ported on a mobile terminal, intercepts the IDs of these local 'radio beacons' (having unique or semi-unique IDs), and then it looks up positions of correlated radio sources in a locally-cached database. Based on that as an input, Place Lab performs a computation akin to triangulation [SLA01] based on which it provides a mobile terminal's position for location-based mobile applications.

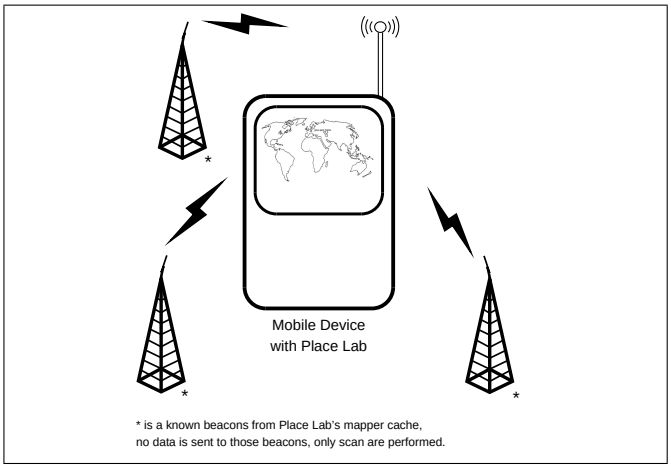


Figure 2.1: A mobile device equipped with Place Lab

Place Lab Architecture

The Place Lab architecture consists of three key elements: radio beacons in the environment, databases that hold information about beacons' locations, and the Place Lab clients that use this data to estimate their current location.
To make Place Lab clients both extensible and portable, client functionality is broken into three logical pieces: *spotters*, *mappers* and *trackers*.
The spotter's task is to monitor the radio interface and share the IDs of the observed radio beacons with other system components.
The mapper provides the geographical location of known beacons. This information always includes a latitude and longitude, but may also contain other useful information, like the antenna altitude, the age of the data, a learned propagation model, or the power of the transmitter. Mappers may obtain this data directly from a mapping database, or from a previously cached portion of a database.
The tracker is the Place Lab client component that uses the streams of spotter observations and associated mapper data to produce estimates of the current user's position. The trackers encapsulate the system's understanding of how various types of radio signals propagate and how propagation relates to distance, the physical environment, and geographical location [DEV01].

Portability

The Place Lab is highly portable and adaptable, however a small part of the code is native to the hardware so even if it can run on large type of devices such as a Laptop, Pocket PC or Smart Phones (as Nokia Series 60), and each model has to be tested with it. A Hardware Compatibility List can be found on Place Lab's website.

Reliability - Coverage

Place Lab coverage depends on input from its users, therefore it is possible to arrive to the area with radio sources for which there are no corresponding radio beacons registered in Place Lab databases. This situation can be solved with help of wigle.net (Wireless Geographic Logging Engine) database, which may have beacons registered for these particular radio sources. The other option is to gather on-spot information about radio sources and their beacons. This process is called *stumbling* and requires a GPS signal receiver and at least one radio source adapter supported by Place Lab (i.e. Bluetooth, WLAN or GSM).

Accuracy

The accuracy of Place Lab location depends on many factors, where the most important ones are the numbers of intercepted beacons, their coverage, the reliability of the database and the position estimation algorithms.
For example, for a wide-area 802.11 WLAN-based positioning system, Place Lab can estimate user's position with a median positioning error around 10 to 40 meters (depending upon the characteristics of the environment, see for example [NPS01] for more information).

Improving Place Lab's accuracy

In a default setup, trackers may use only the data provided to them by the spotter and mapper; however, to improve accuracy of location estimation, it may use extra data like road paths and building locations [DEV01]. As an example, Place Lab includes a simple tracker that computes a diagram made up of two or more overlapping circles from the observed beacons. This tracker uses very few resources, making it appropriate for devices like cell phones. Place Lab also includes a Bayesian particle filter tracker (i.e., a statistics estimation technique based on the observed data sample) that can utilize beacon-specific range and propagation information to predict user's location. While computationally more expensive, the Bayesian tracker provides about a 25% improvement in accuracy and allows Place Lab to infer richer information like direction, velocity and even higher-level concepts like mode of transportation (walking, driving, etc.).
Another way to improve Place Lab's location estimate accuracy is to use combination of diverse trackers, returning location estimate that has the lowest standard deviation.

2.3.3 *GPS*

Introduction

The *Global Positioning System* is a satellite-based positioning system and it consists of a constellation of at least 24 satellites in 6 orbital planes (Figure 2.2) continuously broadcasting radio messages (containing satellites' location, time of day from atomic clocks and a unique ID of the satellite). Ground stations throughout the world monitor the satellites to ensure that their atomic clocks are kept synchronized. The first satellite has been launched in 1978 and the most recent in 2005 [GPS01].

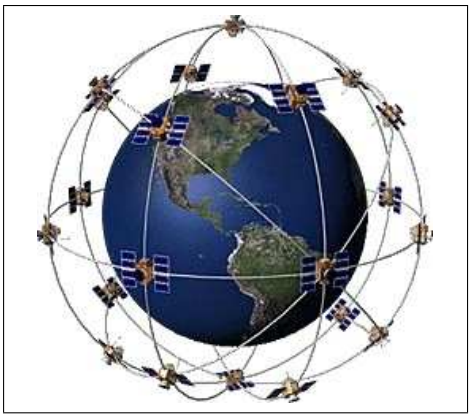


Figure 2.2: GPS Constellation

In our Nomadic Positioning Service, the GPS is used to gather satellites' coordinates such that Place Lab may estimate the device's location through scanning those and matching with known ones after the (earlier) GPS stumbling process.

Determining a position

A GPS receiver knows the location of the satellites through information included in the broadcasted radio messages. By measuring the delay at which messages from different satellites arrive to the GPS receiver, it estimates how far away these satellites are. The estimated receiver - satellite distance determines radius of an imaginary sphere centred at the satellite. The GPS receiver is located where the spheres of (at least three) satellites intersect, see figure 2.3 [GPS02].

Reliability

A GPS receiver requires a clear line of sight to the sky in order to intercept GPS signals. The orbits of GPS satellites are designed such that at least four satellites are always within the line of sight from almost any place on the Earth. Three is the minimal number of satellites signals required for a receiver to calculate its 2D position and four satellites signals is a minimum required to calculate its 3D position and the precise time.

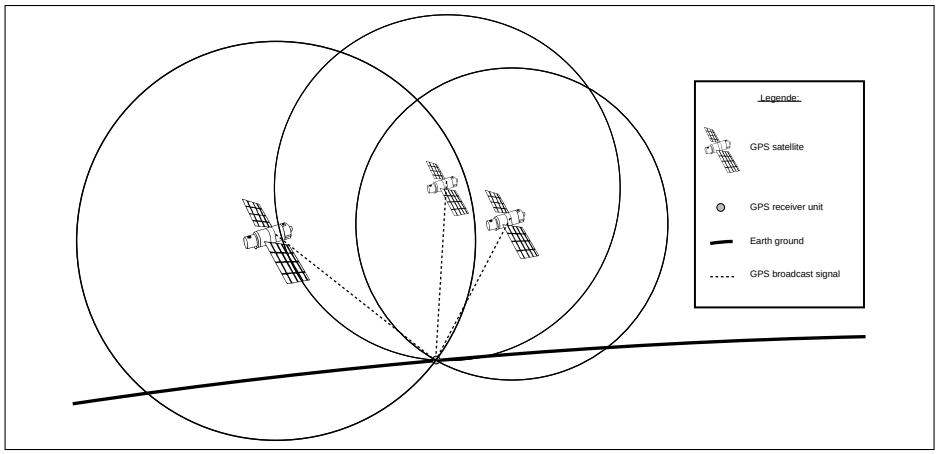


Figure 2.3: GPS determining a position

Accuracy

The accuracy of a position determined with GPS depends on the type of a GPS receiver. Most handheld GPS units have about 10-20 meters accuracy. Some other types of receivers use a method called *Differential* GPS (DGPS) to obtain a much higher accuracy. DGPS requires an additional grounded radio messages sources at a known locations nearby receiver. The DGPS location estimation has accuracy greater than 1 meter [GPS02].

2.4 Time

After presenting Nomadic Positioning Service's location determination methods in section 2.2, this section presents the time determination software and hardware that could be used to determine and then synchronize the Nomadic Positioning Service user's and provider's clocks. The requirements for the integration of time-awareness into the Nomadic Positioning Service are high time-clock accuracy, up-to-datedness (continuous synchronization with a time source) and high clock resolution. To be noted that only an overview of the available technologies for time-awareness will be made, no solution will be designed nor implemented in this assignment; this part will be executed in the parallel student assignment [LBP01].

2.4.1 *NTP*

Introduction

The Network Time Protocol (NTP) is a protocol for synchronizing the clocks of distributed clients' computer systems with a NTP server, over packet-switched, symmetric-latency data networks [NTP02].

Performance

The NTP version 4 can usually maintain time to within 10 milliseconds (1/100 s) over the internet, and can achieve accuracies of 200 microseconds (1/5000 s) or better in local area networks under ideal conditions [NTP02].

Protocol Design Issues

The synchronization protocol determines the time offset of the server clock relative to the client clock. The various synchronization protocols in use today provide different means to do this, but they all follow the same general model. On request, the server sends a message including its current clock value or timestamp and the client records its own timestamp upon arrival of the message. For the best accuracy, the client needs to measure the server-client propagation delay to determine its clock offset relative to the server. Since it is not possible to determine the one-way delays, unless the actual clock offset is known, the protocol measures the total roundtrip delay and assumes the propagation times are statistically equal in each direction. In general, this is a useful approximation; however, in wireless network, each way delays can differ significantly due to packet lost, network overload and others wireless network characteristics [NTP01].

Conclusion

In case of the Nomadic Positioning Service, service provider is mobile and it uses different wireless physical layer such as WLAN, 2.5G and 3G without guarantees of a symmetric uplink and downlink delays. In this case, NTP up-to-datedness (synchronization) may not be reliable, which does not allow for the use of it as a time determination technique for the Nomadic Positioning Service.

2.4.2 *Atomic / Radio clock*

Introduction

An *atomic clock* is a type of clock that uses an atomic resonance frequency standard as its counter. The first accurate atomic clock was built in 1955. This led to the internationally agreed definition of the second being based on atomic time in 1967 by the International System of Units [ATC01].

Performance

An atomic clock resolution can be up to 10^{-9} second. Clock time may be broadcasted and read by distant (mobile) devices through a radio transmission; clocks receivers are called *radio clocks*. Those receivers can get a signal event even in a building and give good time estimation. However, due to the low clock resolution, the precision of an estimated time is low, comparing to other methods.

Conclusion

In case of the Nomadic Positioning Service, we need higher resolution time estimation methods; therefore, radio clock may not be used.

2.4.3 *GPS*

Introduction

The Global Positioning System (mentioned in section 2.2) is a satellite-based positioning system and it consists of a constellation of satellites continuously broadcasting radio signal (containing satellites' location, time of day from atomic clocks and a unique ID of the satellite). This system allow also for time-estimation.

Time Accuracy

The accuracy of a time determined with GPS depends on the satellites and the type of GPS receiver. Both satellites and receivers are prone to timing errors. Ground stations throughout the world monitor the satellites to ensure that their atomic clocks are kept synchronized. Receiver clock errors depend upon the time oscillator provided within the unit. However, clock errors can be calculated and then eliminated once the receiver is tracking at least four GPS satellites.

Time Application

Many synchronization systems use GPS as a source of accurate time; one of the most common applications of this use is that of GPS as a reference clock for time code generators or NTP servers clocks [GPS01].

Conclusion

GPS time is much more accurate than atomic clocks, however the reception of GPS signals requires an open sight of the sky (section 2.2.3 provides more details), contrariwise to a radio clock receivers. As in case of the Nomadic Positioning Service, we require the most accurate, synchronized and high-resolution clock as possible, we select the GPS-based time source as the most appropriate one.

2.5 Service Oriented Architecture

This section presents the *Service Oriented Architecture* (SOA) used by the existing implementation of the Nomadic Positioning Service. Particularly, the Mobile Service Platform (MSP) used by our Nomadic Positioning Service is based on a Jini Network SOA. Generally, SOA allows software components to publish, invoke and discover services on a network. SOA also allows a software programmer to model programming problems in terms of services offered by components anywhere over the network.

In SOA we distinguish a *Service provider*, a *Service requestor*, and a *Service registry* (see figure 2.4). The Service provider is responsible for publishing a description of the service to the Service registry. The Service registry is a service repository and it provides the capability of discovering services by the Service requestors. The Service requestor is responsible for discovering and invoking the service. The Service requestor may bind to the service obtained from the service registry and to be used by it [SOA01].

2.5.1 *Jini Network*

Introduction

Jini network technology provides a flexible infrastructure for delivering services in a network and for creating spontaneous interactions between service provider and their users (i.e. clients), regardless of their hardware or software

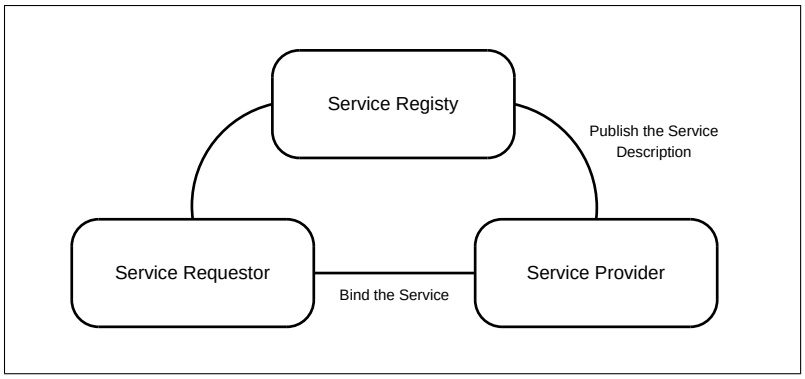


Figure 2.4: SOA Architecture

implementations. Jini technology can be used to build adaptive networks that are scalable, evolvable and flexible as typically required in dynamic computing environments [JIN01].

Architecture

The Jini architecture is composed of a *Lookup Service* (SOA's Service registry), a *Jini Service* (SOA's Service provider) and the *Jini Client* (SOA's Service requestor). Figure 2.5 presents a typical service registration and request procedure.

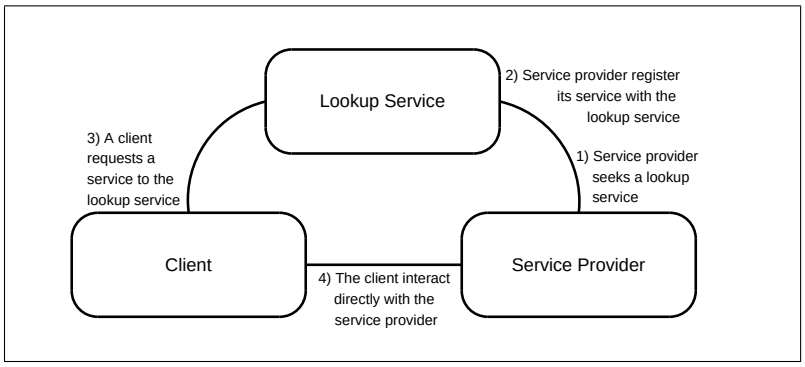


Figure 2.5: Jini Architecture

The Lookup Service (LUS) keeps track of the Jini services and provides the *proxies* to communicate with the service. The LUS is itself a Jini Service as well.

The Jini Services are registered with the LUS and are capable of being invoked through their public interface defined via a Java remote interface. The underlying system that allows Jini services to communicate is a *Remote Method Invocation* (RMI). The Jini Client requests proxy from the LUS in order to invoke particular Jini Service.

2.5.2 *Jini Surrogate*

The goal

In order for a hardware or software component to join in a network of Jini technology-enabled services, it must satisfy several critical requirements. Particularly, it must be able to participate in the Jini *discovery* and *join* protocols and it must be able to download and execute classes written in Java programming language. In addition, it may need the ability to export classes written in Java programming language so that they are available for downloading to a remote entity. In our case, the Nomadic Positioning Service can be executed on PDAs that cannot satisfy those requirements; therefore, the Nomadic Positioning Service cannot participate directly in a Jini network. To overcome that, the Jini Surrogate Project provides a solution with the aid of a third party co-called *Jini Surrogate* for a service 'representative'. It allows the nomadic service to participate in a Jini network, while maintaining the plug-and-work model of Jini network technology [SUR01].

Architecture

The Surrogate architecture is composed of four components: the *device*, the *Surrogate Host*, the *Interconnect* and the *Jini Network*. The term device refers to a hardware or software component that is not capable of directly participating in a Jini network. The Surrogate Host is a framework that resides on the host-capable machine and provides a Java application environment for executing the Surrogate. The Interconnect is the logical and physical connection between the surrogate host and the device.
The Jini Surrogate provides a service gateway that enables limited Java devices to hook into a Jini network, for this the Jini Surrogate has to implement a interconnect protocol which includes the interconnect-specific mechanisms for services discovery, retrieval of the surrogate, and aliveness [SUR02].

The Figure 2.6 presents the role of the Surrogate for the communication between the provider of the service on the mobile device, e.g. Nomadic Po-

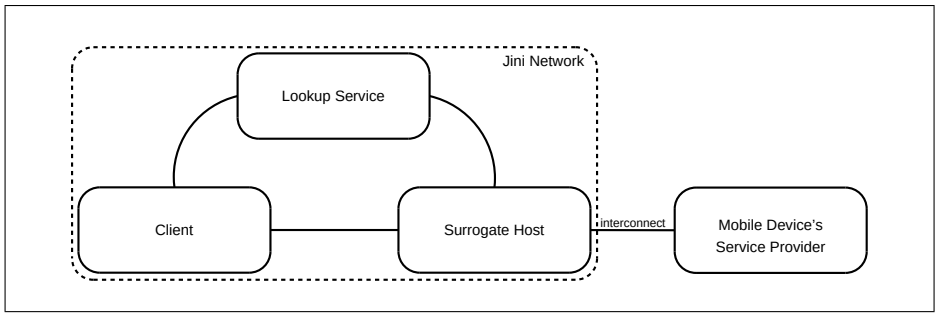


Figure 2.6: Surrogate Architecture

sitioning Service provider, and other components of the Jini Network.

2.6 Nomadic Positioning Service

The implementation of the Nomadic Positioning Service is called *MSP Location Service*; the Nomadic Positioning Service is just its conceptual name.

This section presents the main components of the MSP Location Service and their communication within the Jini network. The MSP Location Service is composed of a *MSP Location Client*, a *MSP Location Server*, the *Surrogate Location Service* (that runs on the *SurrogateHost*) and the *Lookup Server*.

2.6.1 MSP Location Server

The MSP Location Server publishes his position on the Surrogate Location Service. That is because as we explained in section 2.4.2, the server is deployed on PDA platform and therefore cannot meet requirements to be a part of the Jini network.

Figure 2.7 presents the time sequence diagram of the MSP Location Server registration and publishing a service through SurrogateHost and Lookup server. The interconnect protocol implemented by our Nomadic Positioning Service is a HTTP-interconnect which use HTTP request to send message through the Interconnect. It uses two modules one, named IO module, for the mobile device to communicate with the Interconnect and the other, named interconnect module, for the Surrogate to communicate with the Interconnect (which is the logical and physical connection between the Surrogate Host and the device).

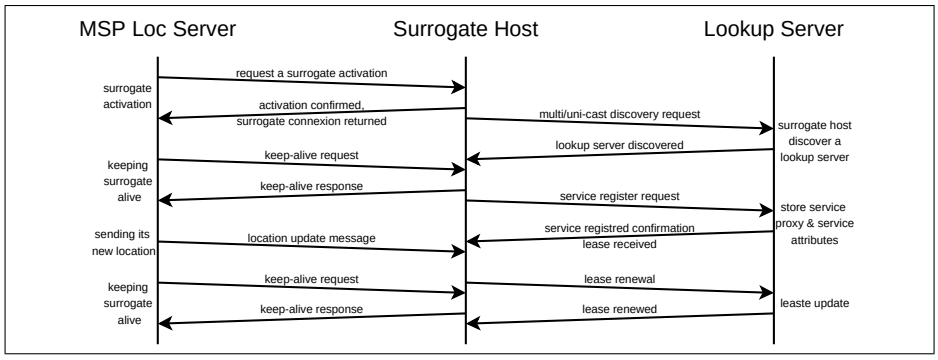


Figure 2.7: Service publishing procedure

2.6.2 Surrogate Location Service

The Surrogate Location Service registers the MSP Location Server's service to the Lookup Server and then it provides Location of the MSP Location Server on the Jini network for the service users.
A Surrogate Location Service is a Jini-based service that runs on the SurrogateHost. Particularly, it allows the mobile device to provide a service on the Jini network. It implements a HTTP-interconnect protocol which allows it to communicate with the Interconnect.

2.6.3 MSP Location Client

The MSP Location Client allows a service user to obtain service, which would mean gathering a mobile device's position. The MSP Location Client can run on a mobile or fixed device, however for now only a J2SE implementation is available.

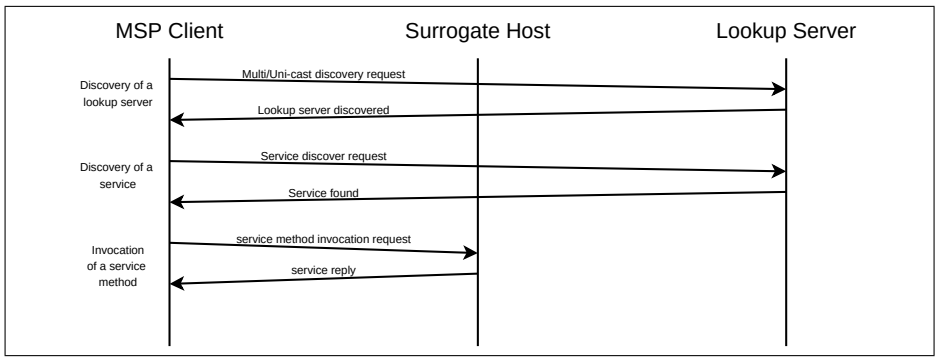


Figure 2.8: Client start-up time sequence diagram

Figure 2.8 presents how the MSP Location Client discovers and invokes the service through the Lookup Server and the SurrogateHost.

2.6.4 LookupServer

The Lookup Server advertises available nomadic services. When a service provider publishes a service, it sends its service proxy-object and service attributes to the Lookup Server, and the latter provides this service object to any client that requests a matching service.

Chapter 3

DESIGN OF NOMADIC TIME-BASED POSITIONING SERVICE

3.1 Introduction

This chapter presents a design of the Nomadic Time-Based Positioning Service. Based on the background research and the study of the existing Nomadic Positioning Service, we propose its redesign such that it integrates time-awareness for the service provider and service users (as we indicated in section 1.4.2).

This chapter starts with an introduction on existing design of the Nomadic Positioning Service; i.e., design of the service before integrating the time-awareness into it. Then we define the set of requirements for the Nomadic Time-Based Positioning Service and based on them we propose a high-level service design; representing service architecture with all internal components and their interactions. The evolution of the existing Nomadic Positioning Service into the Nomadic Time-Based Positioning Service, needs to be done while keeping the existing design features and adding some new features. We achieve that by firstly answering the research questions (indicated in section 1.5).

Figure 3.1 presents the actual design of our Nomadic Positioning Service. Based on the figure, we can further easily identify all the necessary modi-

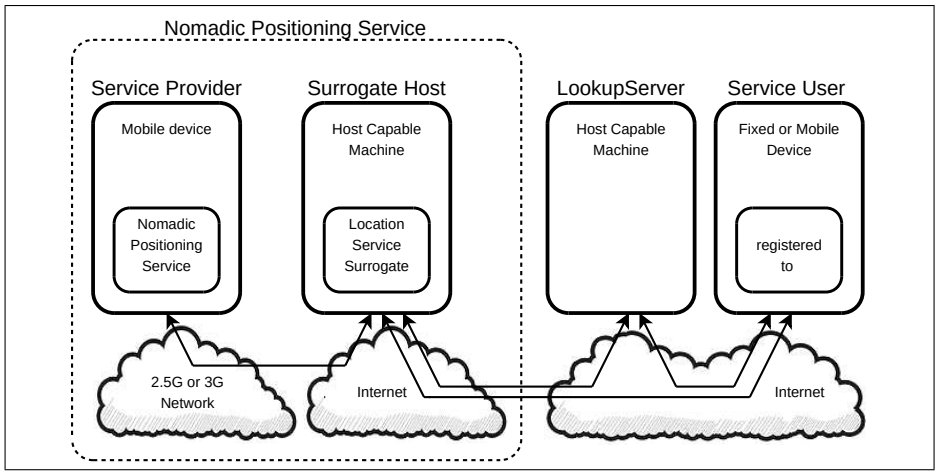


Figure 3.1: Nomadic Positioning Service

fications leading towards the time-awareness of this service. The Nomadic Positioning Service is composed of at least two components: the Surrogate Host and a Service Provider. There can be multiple surrogate instances for each Service Provider, and any surrogate instance is executed in the Surrogate Host. Besides the Nomadic Positioning Service itself, two more components are needed to access and use the service: a Lookup Server to publish the service and a Service User to actually use it. The connection between the Service Provider and the Surrogate Host use the Interconnect protocol implementation of the Mobile Service Platform.
The service that is provided by the existing Service Provider includes only message location including only the location coordinates of the Provider and his movement since last known position. The Surrogate Host registers its service to the Lookup Server by storing there its *service proxy* and *service attribute*, and then it provides Provider's location event to the Service User through the RMI.

3.2 Requirements

In this section, we define the requirements for integration of time-awareness into the existing Nomadic Time-Based Positioning Service, based on the problems definition (section 1.3.4) and the research questions (section 1.5).

3.2.1 *User/Provider Clock Synchronization*

Research question #1: *What is the most efficient way to accurately and reliably synchronize service user's and nomadic service provider's clocks?*

In our background studies, we have made an overview of three possible time-awareness techniques: the NTP, the Atomic Clock and the GPS.

	Advantage	Disadvantage
GPS	-available everywhere on earth -provides position & time -time accuracy of receiver around 10E-7 second	-sight of sky required -specific hardware receiver required
Atomic Clocks	-no sight of sky required	-specific hardware receiver required -time accuracy of receiver around 10E-6 second
NTP	-no specific hardware receiver required	-only accurate on symmetric latency network

Table 3.1: Time-source comparison

The NTP protocol is not accurate enough while implemented over the wireless communication link due to the lack of guarantees of a symmetric delay (see section 2.3.1 for more details). This was the only time-source studied by us, which is software-based and therefore does not require dedicated hardware.

The Atomic Clock would not provide an accurate enough time-source, as required by the Nomadic Time-Based Positioning Service (see section 2.3.2 for more details). Moreover, the only significant advantage of this technique over the GPS-based would be that the radio clock receiver does not require a sight of the sky as for GPS.

The GPS appear to be the most suitable technique for implementing the time-awareness at the service user and provider sides in the Nomadic Positioning Service. Even if GPS requires dedicated hardware, i.e., GPS receiver for mobile devices, its accuracy is much higher than NTP or an atomic clock. As we indicated earlier, most of GPS receivers have time accuracy around 100 nanoseconds (comparing to radio clock accuracy of around 1 millisecond). However, the more significant disadvantage of the GPS-based solution is a weak or even lack of signal reception in building or anywhere, where there is no direct sight of the sky.

As we said while providing the background information on the Time objective (section 2.3), the problem with synchronization of the Nomadic Positioning Service providers and users, will be dealt with in a separate student's assignment [LBP01]. In this section, we just deal with time-awareness, which mean provisioning of timestamps on the service user and provider sides.

3.2.2 *Timestamp integration*

Research question #2: *What is the most efficient way to integrate timestamp in locations events?*

The timestamp of location event and time estimations is one of the key elements of our Nomadic Time-Based Positioning Service with the user/provider time synchronization. This question could be divided in two sub-questions:

1) What time source should be used?
2) How to integrate the time-awareness in the existing MSP Location Service architecture?

Answer 1) Place Lab's API (section 2.2.2) allows getting the timestamp from location estimation. However, after closely analyzing its spotter's source code, we noticed that the time source of this timestamp is different and depends on the type of the spotter used. While using a WLAN spotter, Place Lab will use the device's system clock as a time source through the well-known J2SE version 1.4.2 class "System", (method currentTimeMillis). This method is known for giving an imprecise time depending on current system clock setup. In case when Place Lab uses a GSM spotter, the time source will be based on GSM antenna time clock, if GSM network supports it, and on the mobile device's system clock otherwise.
Hence we conclude that using a GPS-based spotter will guarantee having the most precise time source, because the GPS time will be used (if received),

and the mobile device's system clock otherwise. We indicate GPS as the most accurate time-source that should be used in the Nomadic Time-Based Positioning Service. See [LBP01] for further details.

Answer 2) The most appropriate way to integrate the timestamp in the existing MSP Location Service is to store it in the existing location object holding the location coordinates. This way the timestamp information is sent in the same message as the location coordinates, hence "an efficient bandwidth usage" design requirement is met.

3.2.3 *Time-awareness for Service User*

Research question #3: *What is the most efficient way to log all time-aware location events and provide them on user's request?*

Again, this question we divide in two sub-questions:
1) What is the most efficient way to save all location-time events?
2) What is the most efficient way to provide location-time event to the service user?

Answer 1) To log all location-time events, we proposed development of *TraceStore* service, which will register itself as a listener to the Nomadic location service of every mobile device running the Nomadic Time-Based Positioning Service. Each time a new location-time event is sent by a mobile device, the TraceStore will be notified by the Jini service architecture and it will store this event in its database. All information regarding the location-time event will be stored, namely: the mobile device's ID, the location expressed in 2D coordinates, the movement since the last known location, the location's accuracy and finally its timestamp.

Answer 2) As the location service's client already needs to make use of the Jini architecture, the best way to provide him with the location updates is to use the same architecture as for the TraceStore's service. This way we keep the existing location service design features (see section 3.2.6 for details).

Summarizing, the TraceStore's service must provide to the Service User:
- The list of all known mobile devices providing nomadic location service
- All traces of a particular mobile device for a specified interval of time in the past
- The last known position of a mobile device providing nomadic location service

We provisioned also one additional method implemented by the TraceStore's service returning the ID of the current TraceStore instance. These methods may be useful to identify an instance of TraceStore, if there is more than one TraceStore service instance running in Jini Network.

For implementation details on these methods, see javadoc (JiniTraceStoreService class).

3.2.4 *Including position accuracy*

As we described in the subsection revealing problems of the existing Nomadic Positioning Service (section 1.3.4), the mobile device's location should be known as along with the location accuracy. As in the timestamp integration subsection (section 3.2.2), in this section firstly we define the location information source, then the location's accuracy, then we present how we will integrate this data into the Nomadic Time-Based Positioning Service.

Place Lab allows retrieving the location's accuracy through its tracker. The location accuracy determination is relative to the location estimation algorithm, therefore once the tracker is implemented in the nomadic location service, the Place Lab API can be used to retrieve the location's accuracy.

As for the timestamp integration, the best way to integrate the location accuracy is to store it in the existing location object holding the location coordinates. This way the information is sent in the same message as the location coordinates so the design feature requirements on "an efficient bandwidth usage" is met.

3.2.5 *TraceStore Specifications*

In this section, we define the TraceStore service specifications, which are necessary to meet the design features requirements.

Storage

The TraceStore is required to log and store multiple location-time events originating in the nomadic positioning-time service provider. To meet this requirement we propose use of a database system such as mySQL. We indicate two following reasons of it. Firstly, this type of information storage has the best performance for an open-source database system. Secondly, it allows

to externalize the storage system of the TraceStore, which further allows to share processing resources, such that as soon as the TraceStore will deal with a huge amount of location events data, accessing the information will be much more faster that any other kind of storage system.

Privacy

The TraceStore access should be controlled as soon as it is use for a real application, because the user location-time data is always a privacy sensitive data. This requirement can be met by setting permission control to TraceStore service and allowing access only for a limited, trusted set of users. For the service evaluation purpose, we granted all permissions for everybody.

3.2.6 *Design features preservation*

Research question #4: *What's the most efficient way to redesign the existing Nomadic Positioning Service while keeping its design features such as a) an efficient bandwidth usage, b) a numerical scalable architecture and c) lack of need for additional configuration when new devices come online ?*

This question we answer in sub-questions a) to c).
Answer a) An efficient bandwidth usage is required by the Nomadic Positioning Service because the connection over the wireless (2.5G or 3G) network is, in most cases, relatively expensive in terms of communication resources as in terms of money. Hence, when data containing location information is sent from the mobile device to the Surrogate Host through the interconnect implementation of the Mobile Service Platform, this data are compressed to reduce its size to minimum. That is why we propose to integrate the timestamp and location's accuracy into the existing location object instead of sending additional data over the wireless link (we provided details in section 3.2.2 and 3.2.4).

b) The numerical scalable architecture requirement will be partly conserved by use of the Jini architecture. The TraceStore's service is provided on the Jini network that allows for dedicating a machine only to this service. If a higher level of scalability will be required, running multiple instances of TraceStore to distribute the processing time may be possible with some improvements (see section 3.5.1 for more details).

c) The lack of need for an additional configuration when new devices come online is implemented through the Jini architecture. On the service provider

side, as the TraceStore automatically register as listener to all location service providers available, so this feature is conserved. On the service user side, as long as the service client discovers the nomadic service and the TraceStore's service through a multicast discovery request, this feature is also conserved.

3.3 MSP Location Client Redesign

In our Nomadic Time-Based Positioning Service, the Client should not only to be connected to the Jini Location Service, but also to the Jini TraceStore's Service. This should be made by a multicast request so the TraceStore address would not need to be known directly by the client.

The MSP Location Client should implement the four TraceStore's service methods (section 3.2.3). We need to note that this client will be used mainly to demonstrate and validate the functionality of the Nomadic Time-Based Positioning Service, rather than to be used in a real service. This is because in reality the MSP Location Client should use the MSP Location Server's service or TraceStore's service transparently to the human user.

3.4 High-level Design

This section presents a design for the Nomadic Time-Based Positioning Service based on the requirements and TraceStore's specifications that were indicated earlier in this chapter (section 3.2).

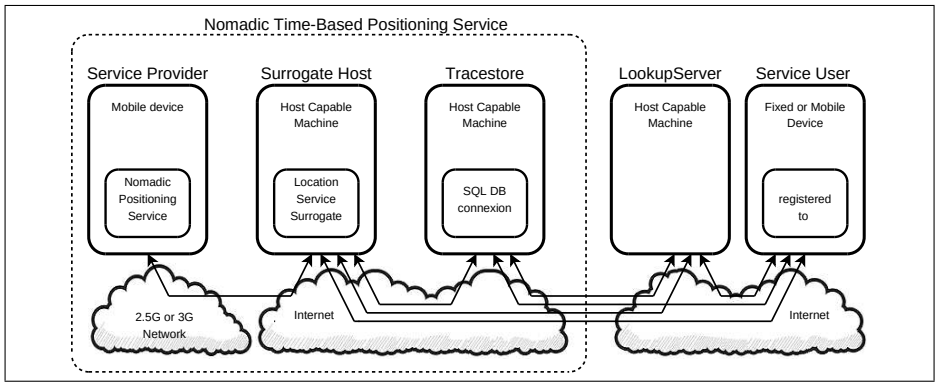


Figure 3.2: Nomadic Time-Based Positioning Service

Figure 3.2 presents a design for the Nomadic Time-Based Positioning Service, which is composed of at least three components: the surrogate host, the TraceStore and the service provider (can be multiple). Most of the possible interactions indicated in the architecture we have already explained in the section dealing with the design of the Nomadic Positioning Service (section 3.1), so in this section we will only focus on a new features and new interactions.
The location-time message that is sent from the Service Provider to the Surrogate Host includes location coordinates, movement since last position, location timestamp and the location's accuracy. The Surrogate Host sends location message to both: the Service User, and to the TraceStore that stores this information and provides it on request (at any time later) to the Service User through RMI.
The Lookup Server publishes the Nomadic Positioning Service and the TraceStore Service, because they register their services to the lookup server by storing there their service proxy and service attributes. Together they provide the Nomadic Time-Based Positioning Service.

3.5 Changes to the existing Nomadic Time-Based Positioning Service

3.5.1 *TraceStore-multiple instances*

The TraceStore should be able to run in multiple instances to allow to the mobile devices' service to subscribe themselves to multiple TraceStore services. This further requires the TraceStore redesign and introducing some features to balance automatically the user load over all TraceStores. Design of this feature is outside of the scope of this assignment.

Chapter 4

EVALUATION

4.1 Introduction

In this chapter, we present evaluation of the design and implementation of the Nomadic Time-Based Positioning Service against its requirements.

4.2 Requirements Evaluation

The minimum set of requirements indicated in the previous chapter (section 3.2) will be evaluated in this section.

4.2.1 *User/Provider Clock Synchronization*

This requirement cannot be evaluated because it was not implemented, as it has been indicated in sections 2.3 and 3.2.1 it will be dealt with in another student assignment [LBP01].

4.2.2 *Timestamp & Position Accuracy Integration*

This requirement has been met. The timestamp and the position accuracy were very well integrated in the existing positioning service, except from the fact that we observed that the position accuracy value is not well estimated by Place Lab's Tracker implementation. Due to some errors in the Place Lab's calculation on the location's standard deviation, the value of the position's accuracy is not valid. This issue is outside the scope of this assignment.

The timestamp and the position's accuracy information are encapsulated

in location-message that is sent from the mobile device through the interconnect protocol to the surrogate. This way the efficient bandwidth usage requirement is conserved.

4.2.3 *Time-Awareness for Service User*

This requirement has been met. The TraceStore provides very well the time-awareness for the Service User. The latter can request previous location of a mobile device through the methods implemented by the TraceStore's service. The service will always return the information if it exists. We observed that TraceStore always returns information to the user even if the size of the requested information is very huge (hundreds or thousands of locations). Each request of past locations from a service user is added in the task queue of the TraceStore service, so if a service user makes thousands of requests, the service will start to treat all the requests from this user first. Because the TraceStore should not get overloaded by one user, this can be a security thread. Hence, we propose that in this particular case, the TraceStore should add the request from this specific user in the queue with a lower priority, especially when there are many simultaneous TraceStore users. This way the TraceStore will serve equally fair for all users even if it is overloaded. Nevertheless the TraceStore has shown that it could stand hundreds of simultaneous requests thanks to Jini architecture and it's Task Manager which allows the treatment of different tasks in multiple thread.

4.2.4 *TraceStore Specifications*

Storage

No particular problems were encountered with the mySQL database (version 4.1.14) used for the service evaluation purposes.

Privacy

As we indicated in section 3.2.5, for the evaluation version of the Nomadic Time-Based Positioning Service all permissions are granted for everybody. For the prototype version of the service, this issue is not very important. However, we indicate that the authentication should be added to the service easily when it is to be used in a real application. This can easily be done by setting specific Jini permissions on the TraceStore service.

4.2.5 *MSP Location Client Evaluation*

The client does not register to the TraceStore service through the same Lookup Server discovery as for the Positioning service, and we are fully aware that this is a shortcoming of our implementation. In our implementation, the Lookup Server discovery request used by the client is only uni-cast, so the client always needs to know the Lookup Server address.

4.2.6 *Design features Preservation*

An efficient bandwidth usage

The bandwidth usage between the mobile device (which provides its location-time events) and the Surrogate Host has been conserved as much as possible. We added only two additional values in the location message; the timestamp and the accuracy of the location estimation. Those two values are then compressed with other data through the MSP interconnect protocol implementation and send as one data to the Surrogate Host.

A numerical scalable architecture

The system scalability has not changed along the implementation of the TraceStore as an additional service. The Surrogate Host can handle many surrogates, but it is necessary to add another Surrogate Host to handle more surrogates if the number of service providers becomes too large, just like for the Nomadic Positioning Service. The TraceStore can be overloaded at its client side or its provider side. At its client side, this would mean that the TraceStore's gets too many location events logs, and on the provider side it would mean that it has to serve too many requests from service users. In both cases, TraceStore should be multiplied in the system, which would require that the TraceStore implements a multiple instances feature (section 3.5.1).

Lack of need for additional configuration when new devices come online

As the client does not discover the TraceStore's Lookup Server through the multi-cast request, but only through a uni-cast request, the TraceStore's address has to be known by the service user. For this reason, this design feature was not kept (details in section 4.2).

4.3 Implementation Problems/Errors Encountered

One of the problems was that the client was not implemented through one lookup server discovery request but by two. First request is needed for discovering the Nomadic Positioning Service and the second for discovering the TraceStore service. We are fully aware that the best way to implement an integrated service would be by using only one lookup request. This error requires the service client knows the Lookup Server address (otherwise, it will not be able to connect to the TraceStore's service). To fix this, the client should connect to the MSP Location Service and the TraceStore Service through the same Lookup method. More precisely, the discovery of the TraceStore service should be integrated in the 'discoverAndLookup' method of the 'Client' class.

Another problem was that due to some errors in the Place Lab's calculation of the location's standard deviation, the value of the position's accuracy is not valid. As soon as another Tracker for the Nomadic Positioning Service is implemented, the 'WiFiTracker' class which contains the calculation of the location's accuracy will have to be re-implemented because the calculation of the location's accuracy depends on the positioning estimation algorithm.

4.4 Conclusion

The evaluation of the Nomadic Time-Based Positioning Service shows that the TraceStore can satisfy the requirements set in the Design chapter even if the current implementation does not satisfy them perfectly. Due to some problems in the implementation part (as mentioned in the section 4.1.2 and 4.1.5), the location's accuracy is successfully sent from the mobile device to the Surrogate but it is not calculated correctly. Moreover, the discovery of the Lookup Server for the TraceStore service does not actually support multicast. Those two requirements were not satisfied in our development, but they can be easily fixed by re-implementing these parts; no system redesign necessary.

Chapter 5

CONCLUSION

5.1 Introduction

In this report, we presented development of Nomadic Time-based Positioning Service that allows the improvement of all application based on the Nomadic Positioning Service by taking into consideration time as an additional dimension. In this chapter we present some value-added services, recommendations for future work and a final conclusion.

5.2 Value-added service

Our Nomadic Time-Based Positioning Service may be used as a standalone service or as a building block in another application. In the first case, an example of a value-added service that enhances the Nomadic Time-Based information is the mapper service developed in [NPS01]. In the second case, it is a service enhanced by the use of the Nomadic Time-Based Service as a component. An example can be an enhanced calendar application that reminds to user diverse events (such as an appointment) in a delay relative to the time needed to get to the event's location.

The knowledge of the past positions of a service provider can allow for better prediction of its average speed, its destination and therefore its next action. It also gives the possibility to know if two service providers have met in location and time. Potentially, almost all existing applications using the Nomadic Positioning Service can be improved by the use of our Nomadic Time-Based Positioning Service.

5.3 Future Work

5.3.1 *Keeping Identification of mobile device through sessions*

An important feature that would enhance the service, would be a functionality enabling a mobile device that goes offline and then comes online again, being identified as one and the same mobile device. In the existing Nomadic Positioning Service, the ID of the mobile device is the timestamp of a moment when it has came online. A better ID would be its MAC address, in this way for a mobile device that was already online in the past and which comes online again, all the location estimation of device's new session would be merged with its old session.

5.3.2 *TraceStore multiple instances*

If the Nomadic Time-based Positioning Service has to be deployed on a very large scale, the multi-instance TraceStore feature should be developed, as mentioned in 3.5.1.

5.4 A portable solution

The Nomadic Time-based Positioning Service offers a full portable solution that can potentially run on almost any device and operating system that support Java standard, Place Lab library and wireless connectivity. Each component has different requirements. The TraceStore components can run on any J2SE platform implementation that can join a Jini Network and that have connectivity to a SQL database. The Service provider has to support J2ME, Place Lab library and wireless connectivity. The Service User, the Lookup Server and The SurrogateHost have to support J2SE and Jini Network connectivity.

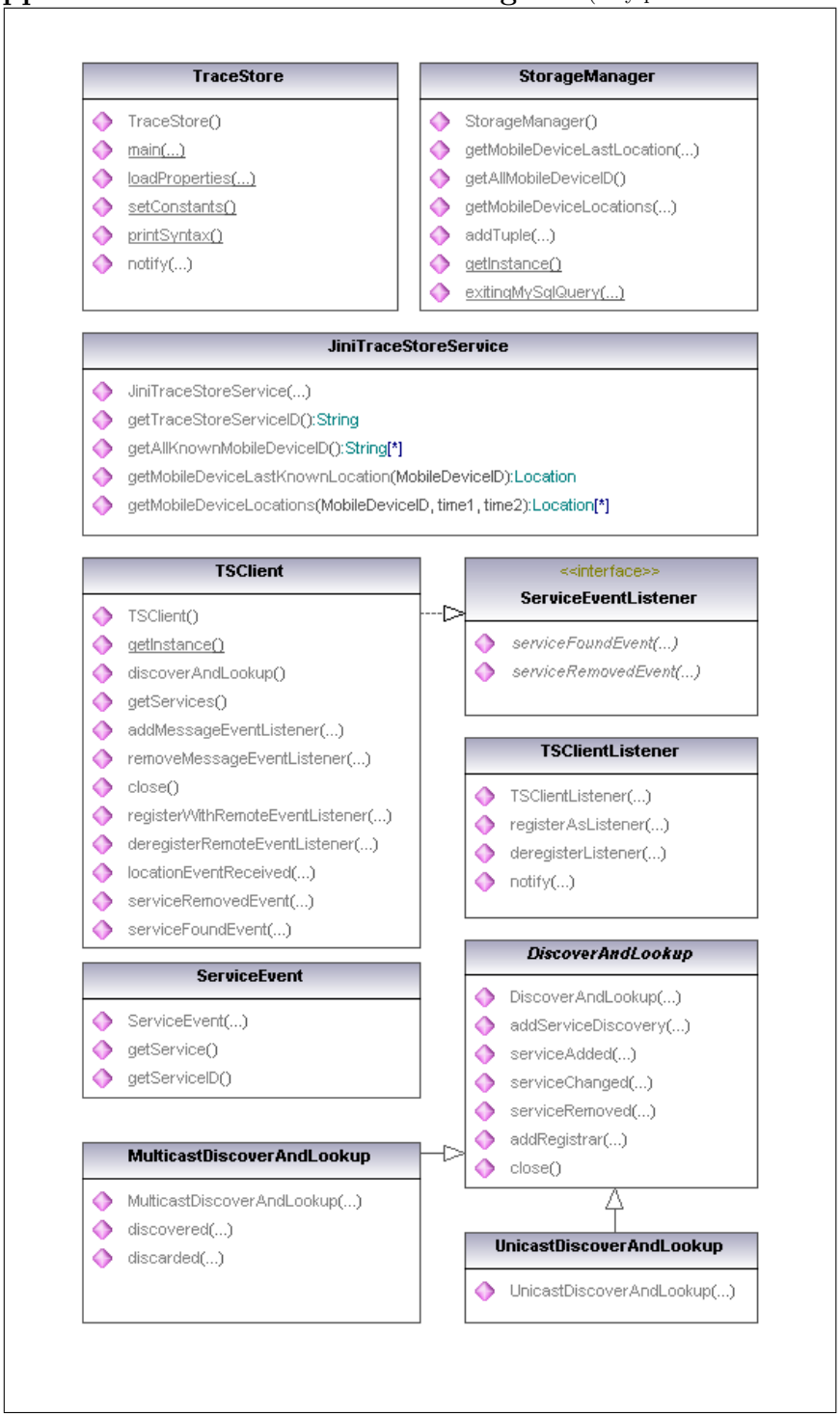
5.5 Conclusion

The goal of this student assignment - the integration of Time-Awareness in the existing Nomadic Positioning Service was successful. All research questions were answered, and all requirements were reached by the design part. The two requirements that were not reached due to their unsuccessful implementation could be fixed easily by a simple re-implementation, no further design is required.

The user and provider time synchronization could not be evaluated because its development is still ongoing; please refer to [LBP01] MSc thesis for more information.

The integration of Time-Awareness in the existing Nomadic Positioning Service enables a new dimension of possibilities in positioning service-based applications. Very valuable information can be deducted from the TraceStore's history, especially if the positioning service based application has linked information with the TraceStore's history, e.g. a list of contacts with their known mobile devices ID.
This new generation of service is possible with Location and Time-Awareness. Further service development based on the Nomadic Time-based Positioning Service is highly encouraged.

Appendix A: TraceStore Class Diagram (only public class are shown)



Appendix B: Nomadic Time-based Positioning Service Source

The source code of this assignment is available through the CVS server.

Readonly access to the CVS repository is possible using the commands below (no password for readonly):

"cvs -d :pserver:anonymous@cvsmsp.ewi.utwente.nl:/local/cvs/msp login"

More information about the ongoing development of the MSP and MSPLocationService is available at the following URL:

http://janus.cs.utwente.nl:8000/twiki/bin/view/MSP/

Bibliography

[AWA01] AWARENESS: Context AWARE mobile Network and Services
http://www.freeband.nl/project.cfm?language=en&id=494

[AWA02] Overview of the AWARENESS project
http://www.freeband.nl/project.cfm?language=en&id=842

[NPS01] Nomadic Positioning Services for a Mobile Service Platform E.A.M. Schoot Uiterkamp, MSc thesis, University of Twente, the Netherlands, 2005

[LBP01] Location Based Performance Measurements, Ger Inberg, MSc thesis, University of Twente, the Netherlands, on going work

[SLA01] a Slashdot article.
http://hardware.slashdot.org/hardware/05/10/04/1418206.shtml?tid=193

[J2ME01] Java 2 Platform, Micro Edition - Wikipedia, the free encyclopaedia http://en.wikipedia.org/wiki/J2ME

[JCP01] Java specification for requests #179
http://www.jcp.org/aboutJava/communityprocess/review/jsr179/

[GPS01] Global Positioning System - Wikipedia, the free encyclopaedia
http://en.wikipedia.org/wiki/GPS

[GPS02] How does GPS work ? http://www.nasm.si.edu/exhibitions/gps/work.html

[PLA01] Place Lab - Quick Start
http://www.placelab.org/toolkit/quickstart.php

[DEV01] Deviceforge.com article
http://www.deviceforge.com/articles/AT8606455669.html

[IRS01] Place Lab publication IRS-TR-05-003
http://www.placelab.org/publications/pubs/IRS-TR-05-003.pdf

[INT01] Algorithms accuracy comparison.
http://seattle.intel-research.net/people/jhightower/pubs
/hightower2004particle/hightower2004particle.pdf

[SOA01] Service-oriented architecture using Jini
http://searchwebservices.techtarget.com/

[JIN01] Jini - Wikipedia, the free encyclopaedia
http://en.wikipedia.org/wiki/Jini

[JIN02] Jini (TM) Architecture Specification
http://www.jini.org/nonav/standards/davis/doc/specs/html/jini-spec.html

[SUR01] surrogate: Surrogate Project.
http://surrogate.jini.org

[SUR02] Jini Surrogate Architecture
http://surrogate.jini.org/sa.pdf

[NTP01] Executive Summary: Computer Network Time Synchronization
http://www.eecis.udel.edu/ mills/exec.html

[NTP02] Network Time Protocol - Wikipedia, the free encyclopaedia
http://en.wikipedia.org/wiki/Network_Time_Protocol

[ATC01] Atomic clock - Wikipedia, the free encyclopaedia
http://en.wikipedia.org/wiki/Atomic_clock

[MPM01] Dimitri Konstantas, Aart Van Halteren, Richard Bults, Katarzyna Wac, Ing Widya, Nicolai Dokovsky, George Koprinkov, Val Jones and Rainer Herzog, "Mobile Patient Monitoring: the MobileHealth System" proceedings of International Congress on Medical and Care Compunetics, NCC, The Hague, June 2-4, 2004