A Web Service based Middleware for Mobile Vehicular Applications

Guido Gehlen, Georgios Mavromatis
RWTH Aachen University
Communication Networks
Kopernikusstr. 16, 52074 Aachen
guge@comnets.rwth-aachen.de

***Abstract*—**
This article introduces a middleware for mobile distributed applications. The platform independent middleware architecture will be presented in general as well as a J2ME realization for mobile phones or PDAs.

Starting from use cases which are motivated from a vehicular research project, a general problem domain of vehicular applications will be analyzed. In this scenario mobile nodes, e.g. vehicles, are connected to a back-end system which wants to monitor data of the mobile nodes, e.g. the location (GPS) data of the vehicles. This class of applications will be addressed by the developed middleware, taking the requirements of the mobile communication link into account and reducing the communication amount over the mobile link.

The implementation is based on the Web Service middleware framework and uses a rule based data filter logic on the mobile node to reduce already on the application layer the communication frequency on the mobile link. The general middleware architecture will be presented with the focus on the rule based data monitoring service. Finally, a communication cost estimation of the proposed architecture will be presented.

I. Introduction

The technological progress in computer and communication engineering evolve more and more to Mark Weiser's vision of ubiquitous computing [1]. Mobile and handheld devices are often more powerful than personal computers of the '90s. Wireless bandwidth is continuously increasing as well as the number of mobile Internet subscriber [2].

To cope with the challenge of designing software for mobile distributed systems a middleware is essential. This middleware should hide the complexity of the distributed system and the communication mechanisms for application programmers. In addition, context information should be provided by the middleware to realize rich context aware applications.

The World Wide Web Consortium (W3C) has published an architectural software framework for distributed systems, called the Service Oriented Architecture (SOA) [3]. The aim of the SOA is to reduce artificial dependencies of interacting software agents. This architecture is applied in the middleware framework called Web Services.

The middleware for mobile devices presented in this paper is based on Web Services. Applications using the middleware are seamlessly integrated into the Web Service network. Mobile devices are able to access any Web Service as well as to provide own Web Services. In particular, vehicular applications could be supported by the middleware, since the flexibility of the Web Service technologies enables the adaptation to the mobile vehicular environment.

The paper will present the Web Service based middleware starting from practical use cases which are covered by the German research project INVENT-VMTL, see section II. After introducing the general architecture in section IV, a rule based data monitoring service will be presented in section IV-B, which reduces the communication amount and costs at the application layer.

II. Use Cases in VMTL

The middleware has been motivated by the German research project *Traffic Management in Transport and Logistics*[1] (VMTL) in cooperation with Ericsson Research Germany. It has been evolved from an application which supports the work of a postman. The overall aim of the project is to optimize the packet delivery chain in respect of reducing the volume of transportation traffic.

The postman is equiped with a Personal Digital Assistant (PDA) and a General Positioning System (GPS) receiver. The PDA reads the data from the GPS receiver for further processing. In the first phase of the project, the PDA synchronizes via Bluetooth with an On Board Unit (OBU) of the delivery vehicle. The OBU enables the connection and interoperation with the backend-system. In the second phase the project addresses also private couriers using their private vehicle without any OBU. Thus, the PDA has to take all of the OBU functions, like connection establishment to the backend-system via the General Packet Radio Service (GPRS), provision of GPS location information, and management of the tour data.

In VMTL different systems from various partner have to work together, thus, it was a common agreement in the project to use platform independent and open standards as well as open source software. These requirements have been satisfied by the use of Web Services, [4]. Consequently, the application running on the PDA has to use

[1] Information available at http://www.invent-online.de

Web Services as middleware framework. From this a general Web Service based middleware for arbitrary mobile distributed applications has been evolved.

First, some aspects of the postman's PDA application will be discussed. The "GPS coordinates" and "tour data" are of interest. The "tour data" consists of a list with all the delivery stops including the customer's contact data and information about the packets. The "GPS coordinates" will be updated frequently by the GPS receiver, the "tour data" will be updated by the VMTL backend-system, and some entries of the "tour data" will be updated by the postman, e.g. after a packet has been delivered the status of the delivery (successful or not) will be updated.

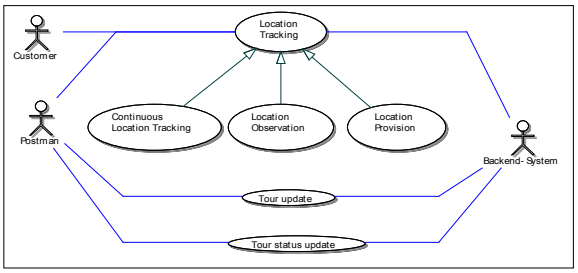

Fig. 1
SOME USE CASES IN THE VMTL PROJECT

Three Use Cases are discussed in the paper. In the first use case "Location Tracking", customers which will be delivered and the backend system want to track the location of the respective packet/transport vehicle on the delivery tour. The second use case describes the update of the tour data on the mobile device, if the tour has been changed or recalculated by the backend-system. The "tour data" consists of list with all the delivery stops including the customer's contact data and information about the packets. In the third uses case the tour data has been changed by the postman on the mobile device. These updates has to be transferred to the backend-system to trigger further processes.

It will be taken a closer look to the first use case "Location Tracking" which is specialized in three use cases, see figure 1. The use case "Continuous Location Tracking" describes the possibility to continuously record the GPS data of the delivery vehicle. In the use case "Location provision" the GPS information will be provided in a service and can be accessed. The use case "Location Observation" enables the observation of the delivery vehicle position.

All of these three specialized use cases have the common ground of an information flow from the mobile PDA to the backend-system. Thus, a communication between these nodes is necessary. The GPS data could be periodically forwarded to the backend-system, the backend-system could poll the location data, or the location information could be filtered at the mobile node and only certain information could be transmitted. This generalized challenge will be discussed in the next section within a simple scenario of one mobile node with a general data base and one backend-system which is interested in the evaluation of this mobile data.

III. GENERALIZED CHALLENGE

For the further discussions this szenario will be generalized and simplified to a szenario of one mobile node, providing arbitrary data, and the backend system which is interested in the data of the mobile node. This szenario is depicted in figure 2.

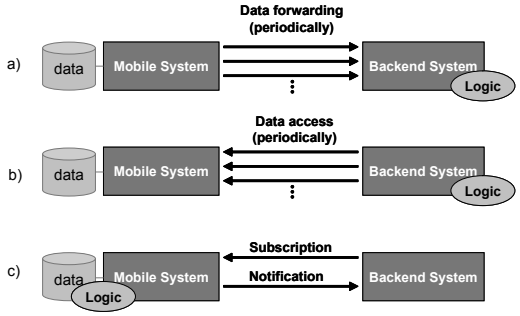

Fig. 2
HIGH LEVEL COMMUNICATION POSSIBILITIES FOR CONTEXT-AWARENESS

Certain variations of the data on the mobile node should activate events in the application of the backend system. If the evaluation logic is running on the backend-system, we have the possibility to periodically poll the data from the mobile node (PDA) or periodically access the data of the mobile node, as depicted in figure 2. But both solutions are cost intensive, since we have to transmit information over the mobile link even if the transmitted data is useless for the backend application.

To minimize the communication frequency over the mobile link, a possibility to displace the evaluation logic to the mobile node is presented. Thus, the data will be locally monitored and evaluated on the mobile node and only relevant data variations cause a notification over the mobile link to the backend system. In other words, the data could be filtered by arbitrary rules, before forwarding it to the backend-system.

Assuming the logic is based on a rule based decision, the rule can be transmitted once from the backend to the mobile system via a subscription. Then the evaluation of the rule starts at the mobile node until a unsubscribe method stops the rule evaluation. Every time the rule applies, the backend system will be notified. For example, an application on the backend system wants to be informed, if a car is in a certain area, as illustrated in figure 6. The backend application has to transmit this rule to the mobile node. Every time the car enters this area, a notification to the backend system is generated.

IV. WEB SERVICE BASED MIDDLEWARE ARCHITECTURE

The overall Web Service based middleware architecture is shown in 3. The middleware is bordered upwards by the

application and downwards by communication layers. The middleware is capable of coupling either to a session layer protocol, like HTTP, BEEP, or WSP, or to a transport layer protocol, like TCP or UDP.

The middleware itself is structured in a protocol part and a service part. The protocol part is based on the Simple Object Access Protocol (SOAP), including the bindings to the underlying protocols and security mechanisms. The service part of the middleware is once again divided into a static part within the U-shaped block and a dynamic part. The static part acts for base middleware functions, like Service Publishing/Discovery and Object Monitoring and Eventing. The Monitor-Service, Notification Service and the Rule Engine are described in section IV-B. The dynamic part depends on the application and adapts on compile or runtime.

In addition, all elements in the service part of the middleware can be distinguished in services and service proxies. A service-proxy is a representative of a remote service and offers the application an interface to this remote service. The service-proxies and the services will map the platform dependent method calls and data objects into platform independent SOAP-calls and vice versa. This architecture bridges the native messaging inside the device environment to the platform independent messaging in the SOA environment.

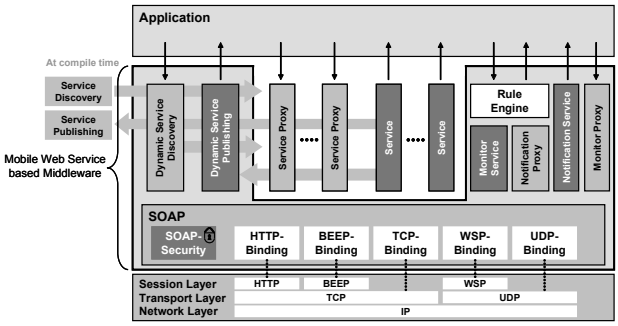

Fig. 3
MOBILE WEB SERVICE BASED MIDDLEWARE ARCHITECTURE

The realization of the middleware has been done for Java enabled mobile devices, compliant to the Java2 Micro Edition (J2ME) standardization. The SOAP part is based on the Open Source libraries kSOAP [5] and kXML [6] and has been extended by additional SOAP-Bindings to alternative underlying protocols and by server capabilities.

The following subsections will present the Mobile Web Service framework in respect of accessing Web Service from mobile devices and providing Web Services on mobile devices.

A. Mobile Web Services

To achieve high interoperability, Web Services has been selected as middleware framework. For messaging the Simple Object Access Protocol (SOAP) [7] is used. It is based on standard internet protocols like HTTP or other arbitrary protocols like the Wireless Application Protocol (WAP) [8] or the Block Extensible Exchange Protocol (BEEP) [9], [10]. The SOAP envelope is structured in Extensible Markup Language (XML). Interfaces are described in a XML subset, the so called Web Service Description Language (WSDL) [11]. This description includes all the information needed in order to invoke service methods from other nodes.

1) Mobile Web Services Access: For the remote access of Web Services a software object (service-proxy) on the device is needed which transforms local method calls, e.g. Java method invocations into SOAP calls. All the information which is needed to generate this service-proxy object is included within the WSDL description of the service. This task is done by a WSDL2J2ME generator which generates with the WSDL information Java code for J2ME and kSOAP.

Most of the existing Internet Web Services (see `www.xmethods.net`) provide a Hypertext Transport Protocol (HTTP) SOAP-Binding. Thus, by default the service-proxy invokes via HTTP POST requests these services. In addition, it has been developed a WAP client binding to SOAP which enables the method invocation in a more efficient way using the WAP, see [8]. The protocol transformation to HTTP is done by the WAP gateway.

2) Mobile Web Services Provision: One of the innovations evolved from the VMTL project is the provision of mobile services on the device of the postman. This enables e.g. the initiation of a tour update by the backend-system. Otherwise, the postman's PDA has to periodically ask for tour updates. The cost saving by using the service on the mobile node is in this use case tremendous, since the probability density of a tour update (p_{update}) is low compared to the frequency of polling the backend-system from the mobile node (f_{poll}).

kSOAP enables by default only Web Service access, but no Web Service provision. We extend the middleware by a lightweight Web- and SOAP-Server [12], [13]. By using this lightweight server, the middleware can publish arbitrary service methods and make these services available via HTTP. In addition, static web content, like HTML pages could be accessed on the mobile device. The HTTP-Server binding for SOAP runs on MIDP1.0/2.0 [14], Personal Java [15], and on any arbitrary standard Java platform. Service classes which should be published by the server have to implement one interface and have to declare their exposed methods and objects to the server.

B. Remote Rule Based Object Monitoring

Based on the mobile SOAP-Server, a Rule Based Monitor service has been developed. The Monitor Service, Notification Proxy, and Rule Engine have to run on the same client as well as the Notification Service and the Monitor Proxy.

The high level communications between the devices are divided into two phases separated in time, see 4. In the first phase the back-end system will subscribe with a rule to the mobile device. The rule is related to the data published by

the mobile device and defines which data changes cause a notification of the back-end system. The second phase is the notification itself. A notification, specified in the rule, is send from the mobile device to the back-end if the rule is fulfilled.

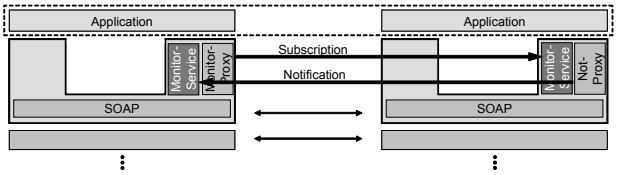

Fig. 4
SUBSCRIPTION TO THE RULE BASED MONITORING AND NOTIFICATION

To realize this concept, a rule parser, a rule evaluator, the subscribe and un-subscribe methods, and a notifier are necessary. The subscribe and un-subscribe methods will add or delete a new rule on the mobile device application. The rule parser will transform the serialized rule into a processable data object, which can be evaluated.

The mobile node offers and publishes a rule based object monitor service with two public methods *Subscribe(Rule)* and *Un-subscribe(RuleID)*. The subscription method contains the Rule and starts at the mobile node the evaluation of this rule. The un-subscription method stops the evaluation of the rule with the corresponding RuleID, see figure 5. The RuleID is assigned by the mobile node after verifying the rule and is returned back to the backend system.

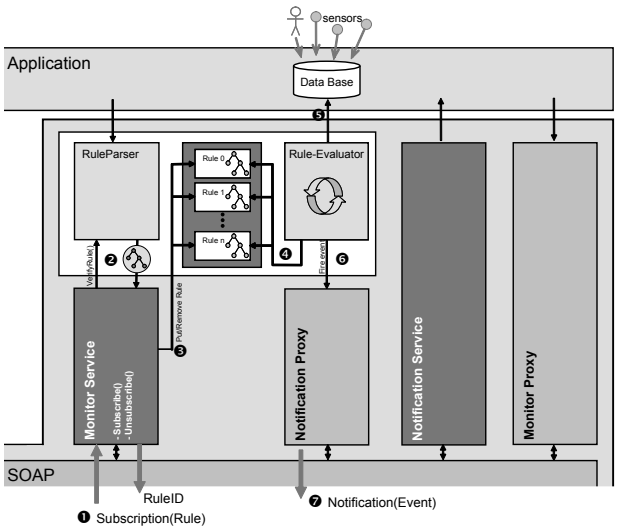

Fig. 5
ARCHITECTURE OF THE RULE BASED MONITORING MIDDLEWARE PART

After the subscription, see (1) in figure 5, each incoming rule is parsed and verified, (2). If the rule syntax is valid, the subscribe method returns a acknowledgment message to the back-end containing a unique Rule ID, else an error message. A valid rule will be saved into an list of rules, (3). In a parallel process this list is sequently evaluated by a RuleEvaluator, (4). Each time the rule applies, a notification service of the back-end system will be invoked with the event as parameter, (7), by invoking the Notification Proxy, (6).

The rules are defined in RuleML. It is a XML based meta language, which defines rules in a platform independent syntax. The RuleML hierarchy of general rules branches into the two direct categories of reaction rules and transformation rules. On the next level, transformation rules are specialized to the subcategory of derivation rules. Then, derivation rules have further subcategories, namely facts and queries. Finally, queries are specialized to integrity constraints [16].

Within this article we solely consider derivation rules. They state a sort of if-then-statements building a filter for the logic-component on the mobile device.

The root element of each rule is the `<rulebase>`-element. Among others, child elements like `<Query>`, `<Fact>` or `<Imp>` are possible. As mentioned before, the main focus in this work lies on implication rules (`<Imp>`) and their structure in order to come up with the demands. The `<Imp>` element consists of a `<head>` and a `<body>` element. Inside the `<head>` element the implication is stated, e.g., send an event to this URL. The condition is nested in the `<body>` element, which is processed recursively. Many conditions are possible, where each condition is declared in an `<Atom>` element. These are coupled by `<And>` or `<Or>` relations, resulting in a complex tree structure. See in the following a example rule description in Rule ML.

```
<Imp>
  <head>
     <Atom>
      <opr><Rel>send</Rel></opr>
      <Ind>Event</Ind>
     </Atom>
     <Atom>
      <opr><Rel>URL</Rel></opr>
      <Ind>137.226.4.133:8080</Ind>
     </Atom>
  </head>
  <body>
    <And>
     <Atom>
       <opr><Rel>lower</Rel></opr>
       <Var>Lat</Var>
       <Ind>5040.0000</Ind>
     </Atom>
        ......
     <Atom>
       <opr><Rel>greater</Rel></opr>
       <Var>Lon</Var>
       <Ind>10020.0000</Ind>
     </Atom>
    </And>
  </body>
</Imp>
```

C. Application costs and reaction time

In this section we will discuss the performance and the costs of applications using this Rule Based Monitoring instead of an straightforward data polling.

We are considering a mobile cellular and packed switched network, like GPRS or the Universal Mobile Telecommunications System (UMTS) with data volume

oriented charging. Thus, the costs (C) are proportional to the transmitted data volume (V). In the case that the rule evaluation logic is running on the back-end side and the poll frequency is f_{poll}, the maximum delay of the application is $\tau = 1/f_{poll}$ and the costs are proportional to f_{poll}. In addition the costs are independent of the probability density that a rule is true (p_{event}). Thus, the costs C_{poll} can be calculated from the data volume V_{poll}, the communication costs per data volume r_V, and the application running time T_{use} to

$$C_{poll} = V_{poll} \cdot r_V \cdot f_{poll} \cdot T_{use} \quad (1)$$

In the case that the logic is running at the mobile side, as in the Rule based Data Monitoring, the application delay depends on the processing time of each rule and the number of rules running on the mobile node, $\tau = N_{rule} \cdot \tau_{proc}$. The costs are now depending on the event probability density p_{event} referring to time ($[p_{event}] = 1/s$).

$$C_{rule} = V_{Not} \cdot r_V \cdot p_{event} \cdot T_{use} + V_{Sub} \cdot r_V \quad (2)$$

V_{Not} and V_{Sub} are the data volumes of one notification resp. subscription message.

Assuming that $V_{poll} \approx V_{Not}$, the cost ratio is

$$\frac{C_{rule}}{C_{poll}} = \frac{p_{event}}{f_{poll}} + \frac{V_{Sub}}{V_{poll} f_{poll} T_{use}} \quad (3)$$

The main and time independent portion of the cost ratio is the factor p_{event}/f_{poll}. Thus, the decision which solution should be implemented in a concrete application is dependent on the application demands on the reaction time ($T_{react,poll} \propto 1/f_{poll}$) and the estimated probability density of an event. For example, if an application demands a reaction time of 1 minute and the estimated event probability density is 1 per hour, the Rule based Monitoring solution is by the factor 60 cheaper.

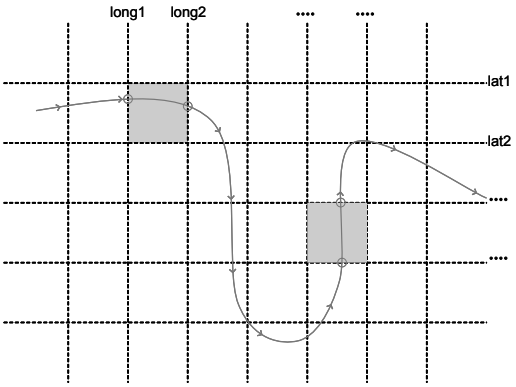

Fig. 6
MOVING VEHICLE THROUGH A MAP WITH A SQUARE LONGITUDE AND LATITUDE GRID

V. CONCLUSION

The paper has presented a middleware for mobile distributed systems in general and an implementation for J2ME mobile devices. The middleware supports the development of mobile applications, in particular for vehicular applications, as illustrated with an application in the research project VMTL. The use of the middleware ease the development process and reduces development and runtime costs.

ACKNOWLEDGMENT

The work presented has been motivated by a cooperation with Ericsson Research Germany within the german BMBF project INVENT-VMTL[2]. The authors would like to thank Prof. B. Walke and Dr. G. Gebhardt of ComNets for their help and support, as well as T. Dinsing and M. Gerdes of Ericsson Research.

REFERENCES

[1] M. Weiser, "The computer of the twenty-first century," Scientific American, 1991. I
[2] W. Mohr, "Trends in mobile communications towards systems beyond imt-2000," ITU Seminar, Ottawa, 2002. I
[3] "Web service architecture," Published on the internet. Available at http://www.w3c.org, 2004. I
[4] R. Wolter, "Xml web services basics," Published on the internet. Available at URL http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/webservbasics.asp, Dec. 2001. II
[5] "Enhydra's j2me soap implementation ksoap," Published on the internet. Available at URL http://enhydra.org. IV
[6] "kxml javadoc," Published on the internet. Available at URL http://kxml.enhydra.org/software/documentation/apidocs/, Feb. 2002. IV
[7] N. Mitra, "Soap version 1.2 part 0: Primer," Published on the internet. Available at URL http://www.w3.org/TR/soap12-part0/, June 2003. IV-A
[8] G. Gehlen and R. Bergs, "Performance of mobile web service access using the wireless application protocol (wap)," in *Proceedings of World Wireless Congress 2004*. San Francisco, USA: University Aachen, Communication Networks, 05 2004, pp. 427–432. IV-A, IV-A.1
[9] M. Rose, "Rfc-3081: Mapping the beep core onto tcp," Published on the internet. Available at http://www.ietf.org/rfc/rfc3081.txt, Mar 2001. IV-A
[10] U. Kefali, "Development and performance evaluation of a simple object access protocol (soap) profile for block extensible exchange protocol (beep)," Master's thesis, 2004. IV-A
[11] R. C. et al., "Web services description language (wsdl) version 1.2," Published on the internet. Available at URL http://www.w3.org/TR/wsdl12, June 2003. IV-A
[12] L. Pham and G. Gehlen, "Realization and performance analysis of a soap server for mobile devices," Communication Networks, RWTH Aachen University, 2004, not yet published. IV-A.2
[13] L. Pham, "Concept and implementation of web services deployed on mobile devices," Diplomarbeit, Communication Networks, RWTH Aachen University, Jan 2004. IV-A.2
[14] "Mobile information device profile (jsr-37), java 2 platform, micro edition, version 1.0," Published on the internet. Available at URL http://jcp.org/aboutJava/communityprocess/final/jsr037/index.html, Sun Microsystems, Palo Alto, USA, Sept. 2000. IV-A.2
[15] "Personaljava application environment specification version 1.2a," Published on the internet. Available at URL http://java.sun.com/products/personaljava, Sun Microsystems, Palo Alto, USA, 2000. IV-A.2
[16] H. Boley, S. Tabet, and G. Wagner, "Design rationale of RuleML: A markup language for semantic web rules," 2001. [Online]. Available: citeseer.ist.psu.edu/boley01design.html IV-B

[2]Information available at http://www.invent-online.de