A Rule based Data Monitoring Middleware for Mobile Applications

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

Abstract— **In this article we look at the scenario of mobile nodes, e.g. cars, which are connected to a back-end system and an application running on the mobile back-end system which wants to monitor data of the mobile nodes, e.g. the position data of the cars. There are many options to realize such an application, but we have to take care of the costs and performance of the application, since a cost-intensive mobile link is used.**

We will present a middleware architecture solving the problem described, and an implementation of this rule based data monitoring middleware by minimizing the communication frequency over the mobile link. The implementation is based on the Service Oriented Architecture (SOA), a middleware framework which tries to simplify as much as possible the software architecture of distributed applications. The realization of our middleware and exemplary application uses Web Service technologies to be platform and vendor independent. Finally, the rule based middleware is evaluated in respect of the application delay and the application communication costs.

I. Introduction

It is considered the scenario, that a backend-system wants to observe data of a remote mobile node, e.g. the observation of the car position (GPS data). 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 a server application, there is the possibility to periodically poll the data from the mobile node (car) or periodically access the data of the mobile node, as depicted in figure 1. But both solutions are cost intensive, since data has to be transmitted 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 middleware is presented which displaces the evaluation logic to the mobile node. Thus, the data will be filtered on the mobile system before transmitting it to the backend-system.

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.

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

The postman is equipped 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 partners 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, [1]. Consequently, the application running on the PDA has to use Web Services as middleware framework, as well. From this a general Web Service based middleware for arbitrary mobile distributed applications has been evolved.

One of the challenges in the research project VMTL was to track the location of the postman efficiently. The PDA of the postman continuously reads the GPS data from the receiver. The backend-system is somehow interested in this data. Thus, the problem is to transmit the GPS data as fast as the application delay admits this. A simple polling of the GPS data may is expensive, since the application has to periodically poll the data with that frequency to achieve the application requirements.

II. Generalized Challenge

For further discussions this scenario will be generalized and simplified to a scenario of one mobile node, providing arbitrary data, and the backend system which is interested in the data of the mobile node. This scenario is depicted in figure 1.

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

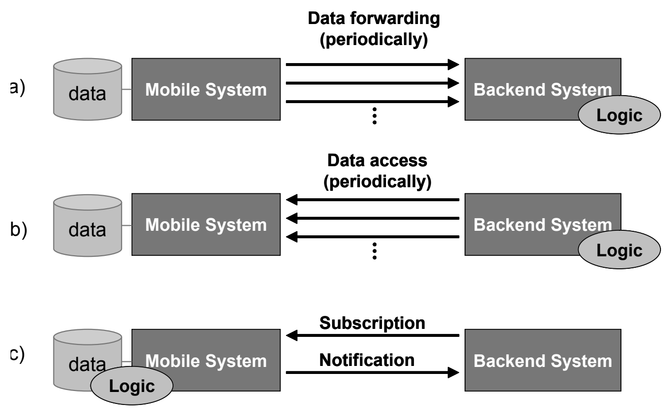

Fig. 1. High level communication possibilities for context-awareness
a) Periodically forwarding of the context data
b) Periodically access of the context data
c) Rule based data monitoring

depicted in figure 1. 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 that 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, see section III-B and figure 5 . 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.

III. Web Service based Middleware Architecture

The overall Web Service based middleware architecture is shown in 2. 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 [2], or WSP [3], 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) [4], 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 III-A. 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 [5].

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 [6] and kXML [7] and has been extended by additional SOAP-Bindings to alternative underlying protocols and by server capabilities.

The server capabilities are realized by developing a lightweight HTTP-server and a server binding to SOAP. The server accepts incoming HTTP requests and passes the SOAP envelope to the mobile SOAP-Server [8] [9]. After validating the content the corresponding methods will be invoked.

A. 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 figure 3. 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.

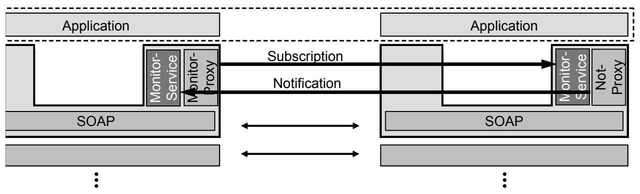

Fig. 3. Subscription to the Rule based Data Monitoring Middleware and notification

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

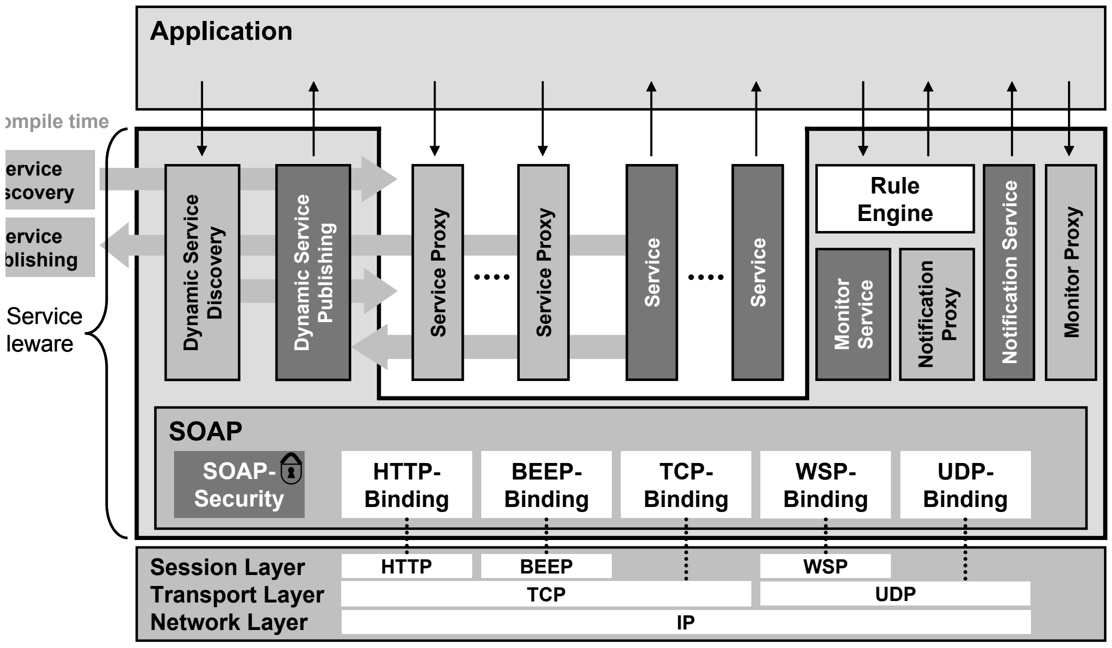

Fig. 2. Mobile Web Service based Middleware Architecture

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 4. The RuleID is assigned by the mobile node after verifying the rule and is returned back to the backend system.

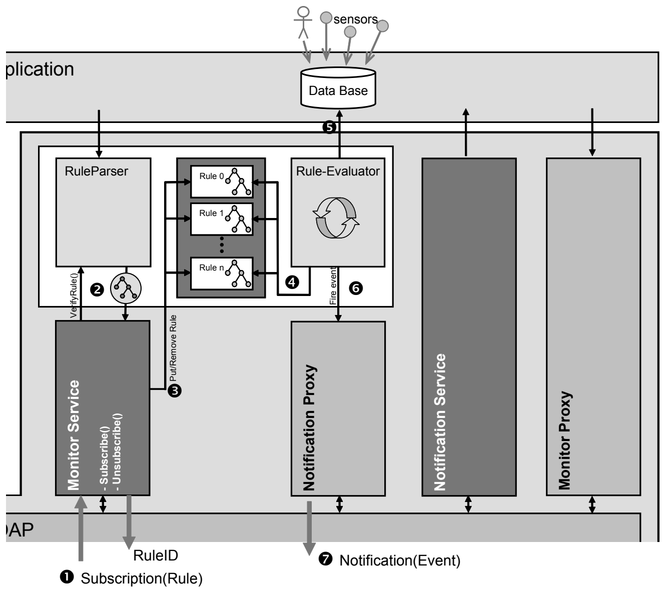

Fig. 4. Architecture of the Rule based Data Monitoring part within the overall Middleware

After the subscription, see (1) in figure 4, 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 Extensible Markup Language (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 [10].

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 RuleML.

```
<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>
```

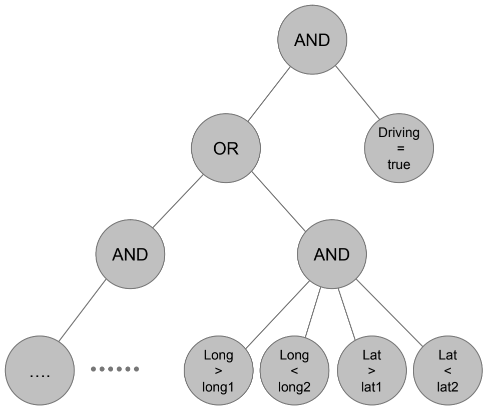

Fig. 5. RuleML Document Object Model tree

B. Application costs and reaction time

In this section the performance and the costs of applications using this Rule Based Monitoring instead of periodical data polling is estimated. 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 of forwarding the data from the mobile system to the backend-system, figure 1 a), or accessing the data from the backend system, figure 1 b), the mobile link is periodically occupied with the volume V_{poll}. Using the Rule based Monitoring Middleware, figure 1 c), at first a subscription is sent and in unsteady time intervals a notification message, see figure 6.

It is assumed that the mobile system permanently observes the data (context value in figure 7), like e.g. the GPS information. The application delay is defined as the time interval between the measurement of the data at the mobile system

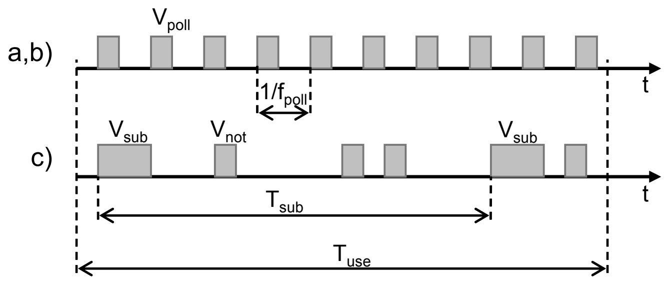

Fig. 6. Communication amount and frequeny

and the reception at the backend-system. In figure 7 the data changes are polled every $1/f_{poll}$ times and received delayed by the processing time τ_{proc} and the network delay τ_{net}.

In the case that the rule evaluation logic is running on the back-end side and the poll frequency is f_{poll}, the maximum application delay, see figure 7, is

$$\tau_{poll,max} = 1/f_{poll} + \tau_{net} + \tau_{proc} \tag{1}$$

The runtime costs of the application within the time period of T_{use} 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

$$C_{poll} = V_{poll} \cdot r_v \cdot f_{poll} \cdot T_{use} \tag{2}$$

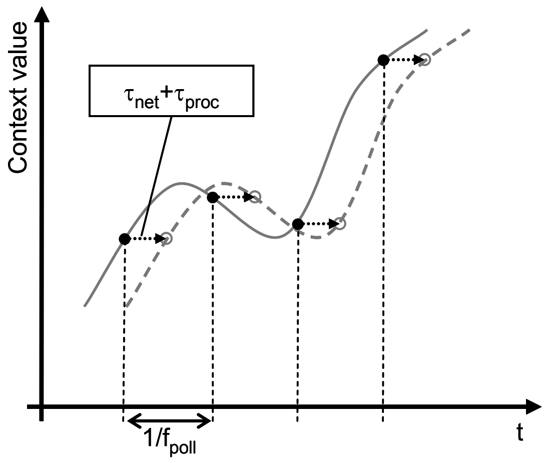

Fig. 7. Delay of polling data from the mobile to the backend

In the case that the logic is running at the mobile side, as in the Rule based Data Monitoring, the application delay depends only on the network delay (τ_{net}), the processing time of each rule, and the number of rules running on the mobile node

$$\tau_{rule,max} = N_{rule} \cdot \tau_{proc} + \tau_{net} \tag{3}$$

The costs are now depending on the event probability density p_{event} referring to time ($[p_{event}] = 1/s$).

$$C_{rule} = V_{Not} \cdot r_V \cdot p_{event} \cdot T_{use} + \left\lceil \frac{T_{use}}{T_{sub}} \right\rceil \cdot V_{Sub} \cdot r_V \quad (4)$$

V_{not} and V_{sub} are the data volumes of one notification resp. subscription message and $n_{sub} = \left\lceil \frac{T_{use}}{T_{sub}} \right\rceil$ is the number of subscriptions within the runtime T_{use}.

Comparing the application delay of option a), b) with option c) by disregarding the processing time and network delay (τ_{proc}, $\tau_{net} \rightarrow 0$), the application delay of option c) is 0 and $\tau_{poll} = 1/f_{poll}$. Thus, the application delay depends only on the polling frequency.

The cost ratio of the runtime application costs in case c) compared to a), b) is

$$\frac{C_{rule}}{C_{poll}} = \frac{V_{not}}{V_{poll}} \cdot \frac{p_{event}}{f_{poll}} + \frac{n_{sub} V_{sub}}{V_{poll} f_{poll} T_{use}} \quad (5)$$

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

$$\frac{C_{rule}}{C_{poll}} \approx \frac{p_{event}}{f_{poll}} + \frac{n_{sub} V_{sub}}{V_{poll} f_{poll} T_{use}} \quad (6)$$

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 concerning the application delay ($T_{react,poll} \propto 1/f_{poll}$) 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 approximately by the factor 60 cheaper.

IV. Conclusion

The paper has presented a rule based data monitoring service within a mobile middleware based on Web Services. The overall architecture and processes have been described as well as the Rule based Data Monitoring in detail. The middleware provides basic functionalities to integrate and provide Web Services and to reduce the communication frequency by applying a subscription mechanism.

The Rule based Data Monitoring enables in many use cases of vehicular application a tremendous communication cost reduction. The application delay is independent of the communication frequency, thus, the costs depend only on the event probability and the subscription update frequency.

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 L. Pham of ComNets for their help and support, as well as T. Dinsing and M. Gerdes of Ericsson Research.

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

References

[1] 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.

[2] U. Kefali, "Development and performance evaluation of a simple object access protocol (soap) profile for block extensible exchange protocol (beep)," Master's thesis, 2004.

[3] 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.

[4] 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.

[5] "Web service architecture," Published on the internet. Available at http://www.w3c.org, 2004.

[6] "Enhydra's j2me soap implementation ksoap," Published on the internet. Available at URL http://enhydra.org.

[7] "kxml javadoc," Published on the internet. Available at URL http://kxml.enhydra.org/software/documentation/apidocs/, Feb. 2002.

[8] G. Gehlen and L. Pham, "Mobile web services for peer-to-peer applications," Jan 2005.

[9] L. Pham, "Concept and implementation of web services deployed on mobile devices," Diplomarbeit, Communication Networks, RWTH Aachen University, Jan 2004.

[10] 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