An Adaptable Framework for Remote Tool Monitoring and Control

Chris Loeser, Robbie Schaefer, Wolfgang Mueller, Marc Borowski
Paderborn University/C-LAB
Fuerstenallee 11, 33008 Paderborn, Germany

Abstract

Engineering collaborations received a new global dimension with the omnipotent access to Internet. Several approaches have been introduced to support communication between collaborating engineers but little attention has been paid to enhance remote access to distributed services from mobile devices. In this paper, we propose a framework for remote tool monitoring and control (RTMC), which can be extended by the user through the definition of application specific plug-ins. The main properties of our RTMC framework are extensibility, interconnection with mobile devices, and independence from any operating systems and devices. To achieve mobile monitoring and control, we applied JXTA, which supports direct communication between stationary PCs and Java enabled mobile phones. We also introduce a so-called user-tracking layer for user location dependent message relay.

1 Introduction

Collaborative engineering constitutes a paradigm of engineering work that is central to the vision of the engineering working environment in the Information Society. From the technical point, most collaborative engineering frameworks do presently not adequately support the integration of very complex engineering environments when interconnecting multiple design groups and do not consider distance-spanning related issues like firewalls, security, remote tool administration, and distributed design flow automation. Distributed engineering development needs however new infrastructures, net-aware tools, and new design methodologies based on re-use in combination with advanced security and network and tool management as they are introduced by the Advanced Collaborative Infrastructure (ACI) in [5, 9], which focus on tool access and design data exchange in engineering environments distributed over several Intranets.

However, existing approaches like ACI mainly consider communication and tool integration aspects in distributed engineering environments. Our current focus is on the monitoring and control of remote engineering tools and services, which can be accessed even from mobile devices. The key idea comes from daily application, where an engineer usually cannot continuously monitor remote tools and services such as complex time and resource consuming engineering simulations. Those services are typically executed remotely and an engineer performs other tasks while the service is running. When such a service has finished or failed, the engineer needs an immediate notification in order to check and to eventually set up a new simulation, if necessary. Especially the case of a simulation run failure needs to be detected as soon as it occurs in order not to waste too much time to set up new runs with new parameters. Reasons for such an abortion can be much different, for example, wrong parameters, disk or system memory overflow, or invalid programs may led to runtime exceptions or segmentation faults. In many cases, the problems can be easily fixed, e.g., by a restart with new parameters or by the use of a different machine in the network. However, the engineer who takes measures typically needs to sit in front of a computer to use, control, and monitor the services. Since some engineering services like complex simulations may run several days, the person could more efficiently spent the time between when notifications are immediately forwarded to home or to business trip locations so that problems can be rapidly fixed.

To support this, we introduce our RTMC (Remote Tool Monitoring and Control) framework. RTMC is an extensible framework that allows to control and monitor tools and services remotely. As we identified the need for controlling these tools in nomadic settings (e.g., on business trips), it is important to provide mobile and secure access, which is addressed by RTMC through a user-tracking layer managing different devices that are used on different locations or occasions. Secure control and access is managed by integration of a TLS layer for encryption and authentication.

The RTMC architecture mainly follows a traditional client/server approach with the tools and services to be controlled at a server side and a monitoring via mobile and stationary clients. We chose the peer-to-peer framework JXTA to implement our RTMC architecture since it is available as source code and greatly supports heterogeneous computing environments with mixed stationary and mobile devices.

The remainder of this article is structured as follows. After elaborating the related work, we introduce basic principles of JXTA. Then we explain the basic RTMC architecture and give an overview of the user-tracking layer before closing the paper with a conclusion.

2 Related Work

RTMC may be considered as a high level and secure alternative to the simple network management protocol SNMP (Simple Network Management Protocol) [10]. SNMP is typically runs a client server application based on the UDP service of the TCP/IP protocol suite. SNMP version 3 (SNMPv3) was defined and implemented specifically to add security to network management. Since version 3 SNMP also supports authentication and encryption. There are a multiple tools available for developers to easily integrate SNMP management information bases (MIBs) and agents into mobile client application software. Multiple tools are available from different vendors such as NuDesign Team [7], MG-SOFT [6], FutureSoft [2], and DMH [1] with different application development environments. Most tools allow defining the MIB elements using a specific syntax that fully defines object attributes.

For remote monitoring on mobile devices, ELC technologies has just released PiranhaWAP [8], an open source WAP/WML-enabled server status reporting product. PiranhaWAP allows displaying real-time system information such as uptime, load average, and memory information on WAP/WML enabled devices such as mobile phones. However, their concept is not extensible and the user still has to deal with the predefined application control capabilities.

3 JXTA

Our approach for remote tool monitoring and control is based on the client/server paradigm, which means that hardware and software resources and services like simulators are on a dedicated tool server and remote clients are used to control those tools. Therefore, at a first glance, it seems to be inappropriate to apply a peer-to-peer framework in order to build such architecture. However, we think that JXTA [4] is a good alternative for implementing a remote tool monitoring and control framework.

Project JXTA is an open-source project initiated by Sun Microsystems as a thin alternative to their Jini framework [3]. It has been designed with the participation of a growing number of experts from academia and industry. Project JXTA was initiated to standardize a common set of protocols for building peer-to-peer applications and to overcome different variations of peer-to-peer APIs and protocols, which have been designed to be implementable on any device. JXTA protocols create a virtual network on top of the existing physical network infrastructure based on which services and applications are set up. This virtual network layer is designed to be as simple as possible with powerful primitives in order to maintain interoperability between different devices and operating systems.

The main purpose of the JXTA virtual network is to hide the complexity of the underlying physical network topology, and provide a uniform addressable network for all peers in the network. The virtual network allows a peer to exchange messages with any other peer independently from its network location. Messages are transparently routed potentially traversing firewalls or NATs (network address translation) with the ability to use different transport/transfer protocols (TCP/IP, HTTP). JXTA standardizes the way in which peers discover each other, self-organize into peer groups, advertise and discover network resources, communicate with peers, and monitors their reachability. The network transport layer is built of a uniform peer addressing scheme based on a peer ID, relay peers that relay messages between peers, and a binary message format to transport binary and XML data.

A peer is uniquely identified by its peer ID (128 bit UUID), even if it uses different network addresses even when the device may have different IPs at different times (e.g., a notebook). Similarly, a device supporting multiple network interfaces (e.g., Ethernet, WLAN, Bluetooth, IrDA) can be addressed as a single peer. The peer ID abstraction allows a peer to encapsulate not just physical transports, but also logical transfer protocols like HTTP and/or secure protocols like TLS and SSL.

JXTA runs on workstations, PCs, PDAs, mobile phones (with MIDP), and web pads and is therefore a good candidate for establishing mobile monitor and control applications. Since JXTA defines a set of XML protocols, the data transfer between the controlling client and the server side tools is managed in a transparent way, allowing easy integration of new tools and services. The independence of programming languages, system platforms, service definitions and low level network protocols - such as TCP/IP or Bluetooth - clearly support our remote monitoring applications. JXTA is able to deal with firewalls or with 'hidden' nodes from network address translation. Therefore, mobile applications are enabled to remotely control tool servers, which are in a secured Intranet. This of course requires secure transmissions of data.

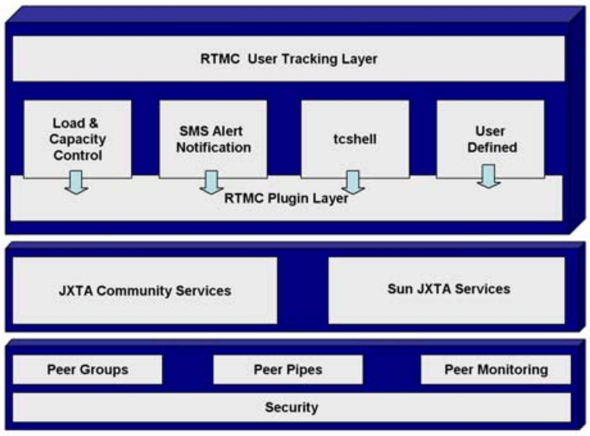


Figure 1: RTMC Architecture based on JXTA

4 RTMC

With our RTMC (remote tool monitoring and control) framework we introduce a system, which seamlessly combines real-time communication and remote service monitoring and control. It includes a notification service when remote services/tools like simulations have completed their execution by successfully finishing or failure. Users may have full control to configure and (re)start service applications remotely. For supporting a distributed architecture of interrelated tools and their respective servers, we consider tool servers as nodes in the network. Each peer is able to gain resource and status information of other peers, which identify themselves as tool servers. Simultaneously each tool server offers this information to all other nodes.

By using and extending the JXTA peer-to-peer framework, we are able to control and monitor those servers as peers, even with mobile devices. JXTA ensures that they are aware of each other by utilizing so-called rendezvous servers, which act as a directory server storing references of peers content and services. To handle limited devices like J2ME enabled mobile phones, JXTA utilizes relay servers, which act as an intermediate translation instance. Because many limited devices are not able to process XML messages, the relay-server transforms standard XML messages into binary XML messages, which are relayed to the limited devices and vice versa.

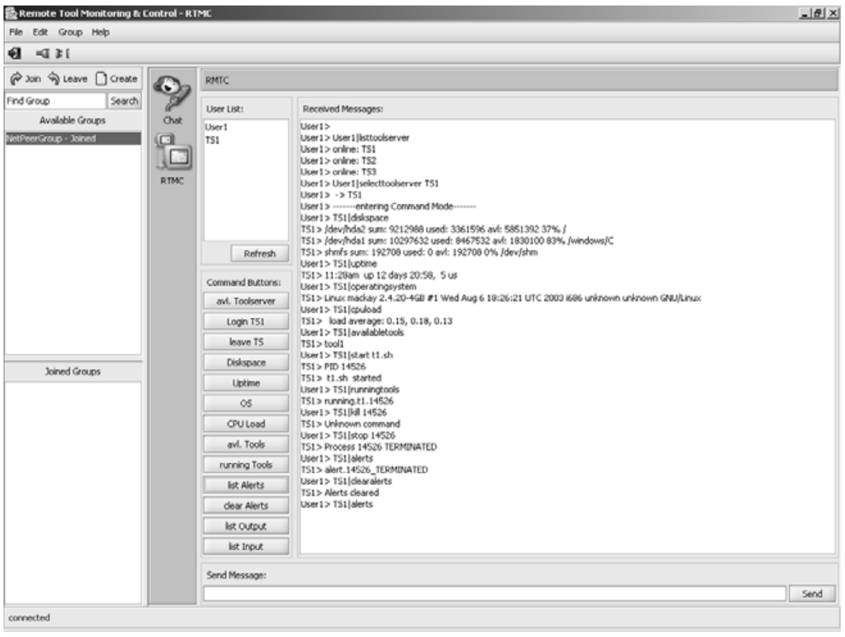
Since JXTA has no built-in support for reliable connections, we extended the protocol in order to keep a connection between two peers, which means that we are able to establish a reliable two-way communication between the tool server and the controlling application. To this end, we implemented a socket-like bi-directional connection that support three different policies for packet exchange: (i) event based (using a listener), (ii) synchronous (wait until it receives a packet), and (iii) asynchronous (received packets are stored in a buffer). Sockets can be connectionless, connection-oriented, reliable, or unreliable.

We built the RTMC communication layer upon the extended JXTA communication, as given in Figure 1. The layer is separated into the plug-in layer and the user-tracking layer. The plug-in layer is for controlling and monitoring capabilities and flows for remote applications and thus provides the core RTMC functionality. It provides a basic set of RTMC functions and defines which and how information is passed to the user or to the controlled application.

An RTMC default plug-in mainly provides means to retrieve system information from a peer and to deliver commands to remote peers. For example, functions can be defined to check for the available disk space, uptime, OS swap activity, available/running applications, alerts, process IDs, and for individual CPU utilization with process re-nicing. Most powerful is the integrated shell plug-in to support user-defined commands and parameters to start tools and to control the remote peer provided the user has access permission to that peer. Additionally, for remote monitoring, we have defined an SMS plug-in to notify a user when an alert occurs. For this, rules have to be defined when a notification alert is sent, e.g., only high priority errors/completions are filtered and sent via SMS while others are just collected in a local logfile.

JXTA as a core system for RTMC has the great advantage, that is comes an integrated instant messaging functionality (in fact one of the most common peer-to-peer applications), which can be used even on mobile devices. Though some manufacturers have already extended SMS to a proprietary SMS chat protocol, those protocols neither work with mobile phones from other manufacturers nor is it possible to interact with other devices like PDAs or PCs. Thus, currently only JXTA provides instant messaging over stationary and mobile devices.

For an enhanced nomadic application consider the following scenario. An engineer starts a time-consuming simulation just before leaving to home in order to have results available on the next morning. Via RTMC services the engineer can receive a message when the simulation stopped for some reason. Via RTMC he/she can retrieve basic resource utilization information in order to get first information about potential problems like disk overflow. When he/she cannot fix the problem from remote, he/she may wish to contact the emergency system administration via instant messaging, email, or telephone to outline the problem. After fixing the problem, the service can be easily relaunched via RTMC.



```
User1>
User1> User1|listtoolserver
User1> online: TS1
User1> online: TS2
User1> online: TS3
User1> User1|selecttoolserver TS1
User1>  -> TS1
User1> -------entering Command Mode-------
User1> TS1|diskspace
TS1> /dev/hda2 sum: 9212988 used: 3361596 avl: 5851392 37% /
TS1> /dev/hda1 sum: 10297632 used: 8467532 avl: 1830100 83% /windows/C
TS1> shmfs sum: 192708 used: 0 avl: 192708 0% /dev/shm
User1> TS1|uptime
TS1> 11:28am  up 12 days 20:58,  5 us
User1> TS1|operatingsystem
TS1> Linux mackay 2.4.20-4GB #1 Wed Aug 6 18:26:21 UTC 2003 i686 unknown unknown GNU/Linux
User1> TS1|cpuload
TS1>   load average: 0.15, 0.18, 0.13
User1> TS1|availabletools
TS1> tool1
User1> TS1|start t1.sh
TS1> PID 14526
TS1>  t1.sh  started
User1> TS1|runningtools
TS1> running.t1.14526
User1> TS1|kill 14526
TS1> Unknown command
User1> TS1|stop 14526
TS1> Process 14526 TERMINATED
User1> TS1|alerts
TS1> alert.14526_TERMINATED
User1> TS1|clearalerts
TS1> Alerts cleared
User1> TS1|alerts
```

Figure 2: RTMC Graphical Front-End for Stationary Machines

Figure 2 shows a screenshot of graphical RTMC front-end, as it is presented to the RTMC user. The upper left has a list of users, which can be contacted for online communication. Messages are entered in the text input field at the bottom. On the left side there is also list of available RTMC commands, where the right hand side displays all events and messages received by RTMC and JXTA, respectively. Through this interface, every RTMC user of the same group has access to exactly the same information, which is a great aid to collaborately solve problems.

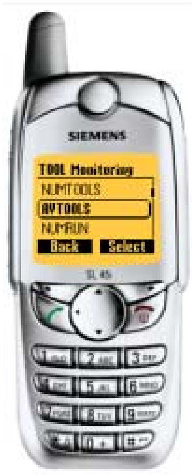


Figure 3: RTMC on a MIDP enabled Siemens SL45i

While this front-end can be used for workstations and PCs, mobile implementations are more limited. Entering lengthy command strings, for example, is feasible but not always very efficient. Therefore, RTMC on mobile phones comes with buttons rather than text input fields whenever possible in order to send RTMC commands. Figure 3 shows an example of the RTMC user interface on a J2ME-enabled mobile phone.

5 UTL - User Tracking Layer

In distributed and mobile environments we face the situation that nomadic users very likely operate with different devices on different occasions and locations. It is not very efficient to broadcast messages to all devices of each user. The critical point, beyond costs and bandwidth, is the filtering of self-created 'spams' since the replication may send multiple copies of the same message. Thus, when a user returns to the PC, he/she does not want to process the same messages again, which were already received by mobile phone or PDA. To tackle that problem, we have defined the user-tracking layer (UTL) for RTMC. The UTL overcomes the issues of device cluttering by managing all the devices of one user.

The core component of UTL is the instance manager, which is added to a JXTA relay server (see Figure 4). When the relay server receives a message, it determines the currently registered device of the receiver. The currently active devices are determined by timeouts. When, for example, the user logs in at the home or at the of-

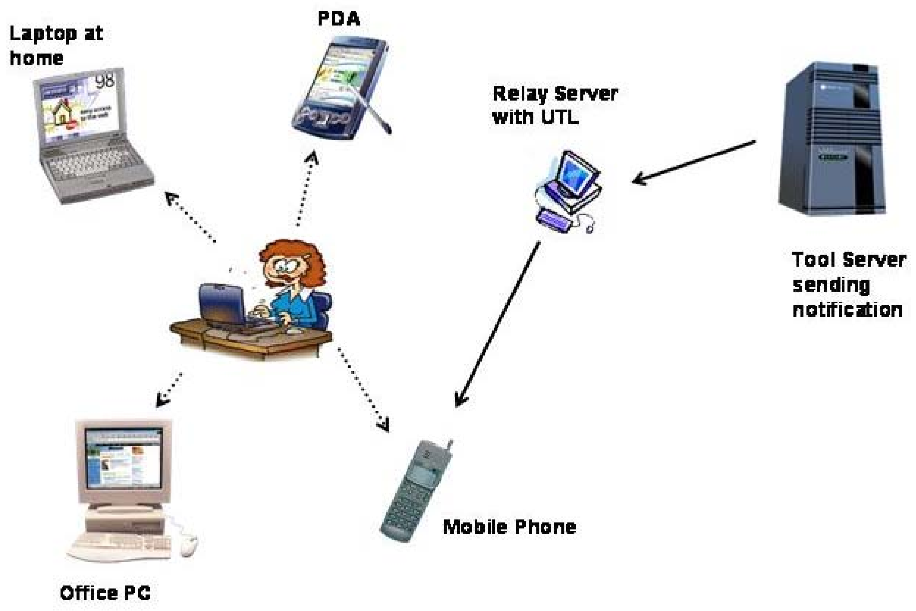


Figure 4: User Tracking Layer - Message Relay to the Active Device

fice PC, the RTMC instance registers the user at the relay server. When the user either logs off or after a predefined timeout, the system tries to relay all messages to the user's default device.

Figure 5 shows a sequence diagram with details of the above sketched behavior of the instance manager, i.e., when a user connects with instance *A* and a second client *B* is already connected. At the beginning, the active instance controller looks for other active instances and decides on the priority of the active client. The priority depends on the status (online, offline) and on the time expired since the last user interaction. After having set the status from client *A* to *online*, the active instance controller compares its own priority with that of other devices (in that case *B*) and detects that its priority is the highest one and thus marks its own instance as *active*. After that, the ping service starts sending ping messages to other clients. The ping messages have information about the active client, time of the next ping, status, and priority. Since all clients receive ping messages, the still active instance *B* is notified as well and recognizes that it does not keep the highest priority. Finally, *B* sets itself to *inactive*. If the new active instance *A* stops sending the ping messages for some reason, instance *B* detects this after the timeout interval and takes the role of the active instance again.

6 Conclusion

In this paper we introduced the RTMC framework, which supports monitoring and control of services in the context of engineering applications. Users can easily define their own application specific control modules, which are loaded into the RTMC framework as plug-ins. Due to JXTA, RTMC supports even small computing devices like mobile phones with instant messaging support. With a user-tracking layer, data can be sent to the currently active or to the default device of a user.

Though we have developed RTMC in the context of engineering applications like remote control and monitoring of simulators, we see a great potential for application in remote maintenance since, for instance, maintenance personal could easily download error codes via RTMC to mobile phones once the remote device is connected to a network.

Acknowledgement

The work described herein was funded by the IST project E-Colleg (IST-1999-11746).

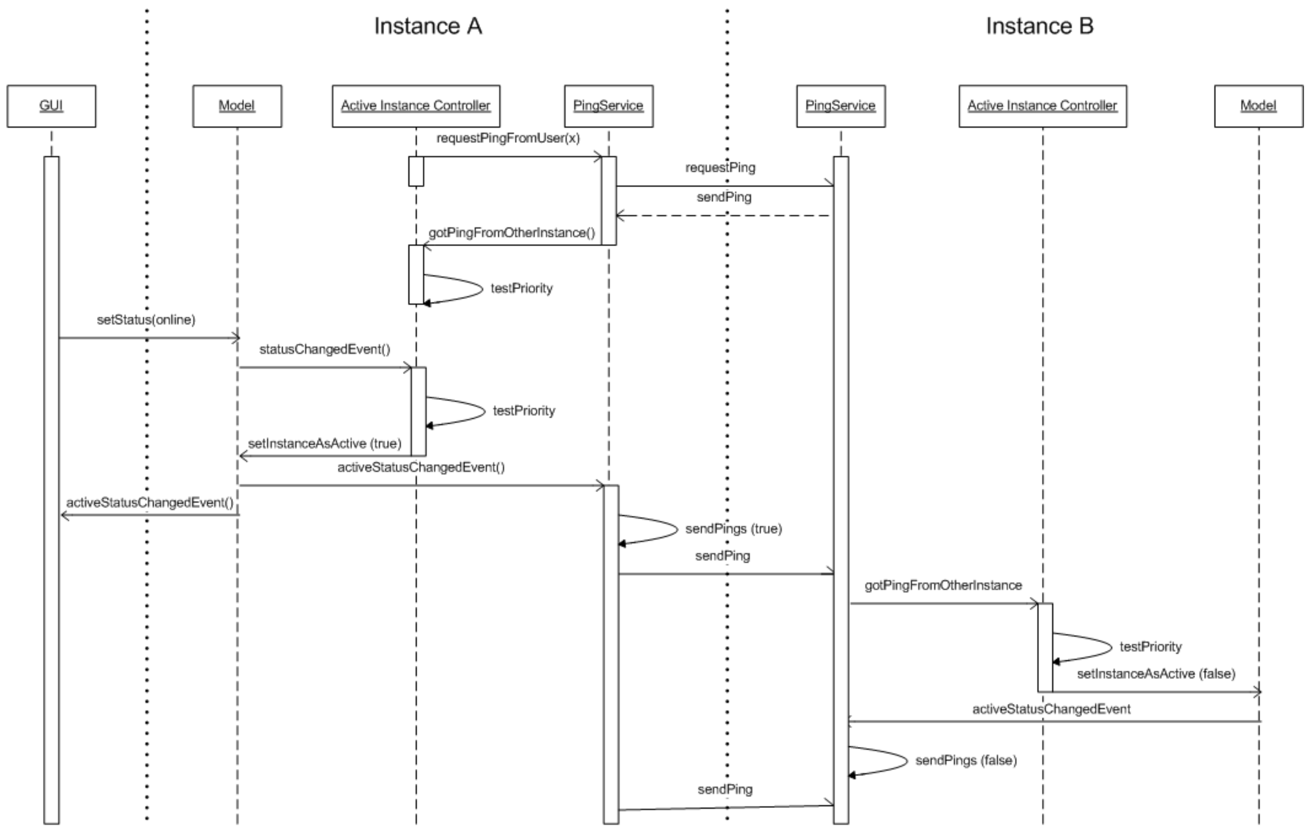


Figure 5: Sequence Diagram of the Instance Manager

References

[1] DMH: www.dmhsoftware.com

[2] FutureSoft: www.futuresoft.com

[3] Oaks, S. ; Wong, H.: JINI in a Nutshell. O'Reilly Verlag, Köln, 2001.

[4] Project JXTA, Sun Microsystems: www.jxta.org

[5] Kostienko, T.; Mueller, W.; Pawlak, A.; Schattkowsky, T.: An Advanced Infrastructure for Collaborative Engineering in Electronic design Automation. CE 2003, Madeira, Portugal, July 2003.

[6] MG-SOFT: www.mg-soft.si

[7] NuDesign Team: www.nudesignteam.com

[8] PiranhaWAP, ELC Technologies Inc.: www.elctech.com/ products_piranha.shtml

[9] Schattkowsky, T.; Mueller, W.: Distributed Engineering Environment for the Design of Electronic Systems. CCE'03, April 15-16, 2003, Poznan Poland.

[10] Stallings, W. B.: SNMP, SNMPv2, SNMPv3 and RMON 1 and 2. Addison Wesley Longman Inc., Reading, Massachusetts, 1999.

[11] Thronicke, W.; Fox, W.; et al.: From Tool Integration to Workflow Management - A Lean Integration Solution. In Proc. 2nd World Conference on Integrated Design and Process Technology, Austin, TX, Dez. 1996.