System for Remote Monitoring of Engineering Services

Pavel Ocenasek[1, a *] , Roman Trchalik[2, b]

[1,2]FIT, Brno University of Technology, Bozetechova 2, 612 66 Brno, Czech Republic

[a]ocenaspa@fit.vutbr.cz, [b]trchalik@fit.vutbr.cz

Keywords: Monitoring, system, service, engineering, agent, server, client.

Abstract. This paper presents the concept of monitoring system that provides active periodic monitoring of servers and their specific services and supports variety of notification methods exceeded the monitored value. In addition, it displays statistics and graphs of monitored values and provides monitoring of system resources.

Introduction

Monitoring activities in engineering can be divided into two groups: active and passive, where each has its advantages and disadvantages. [2]

In case of passive monitoring, we do not send any additional data with the existing traffic, so we does not increase traffic in the communication line. This makes testing more accurate than the active monitoring. Passive monitoring is generally based on the capture of asynchronous events. Such event can be for example a SNMP trap notifications from applications (such as a mail server) or NetFlow data. Another way is to monitor the standard syslog log files. The disadvantage of these methods is that we depend on data we receive. Therefore we are not able to test inactive elements. [2]

For the active monitoring, it is required to dispatch a special packet (test). We dispatch the packet using and we find find out the availability and response time of the device. This another method may, in the case of imprudent configuration (means particular individual testing intervals) began to overwhelm the line and thereby cause inaccuracies in the measurement. We can use the SNMP messages, ICMP packets [4] or simple querying service Telnet on the TCP ports. [3] These queries can be performed at any time and thus we have the opportunity to test the availability regularly in the given interval. This independence from the normal operation means that we can test and inactive devices where currently there is no communication.

System Design

The entire system should consist of several parts. Those are client serving as the user interface of the system, the server part for carrying out the monitoring database, which will work with the server part of the system, and optionally the agent gathering system information of the monitored computer.

Because it is a cross-platform monitoring system, the implementation language of Java was selected. This is a simple object-oriented language, which ranks among the most popular programming languages in the world. Due to the use of the latest techniques, applications are being developed in the JDK version 8.

Client

The first part is the client is used as the only system-wide user interface. The user will be able to use it to configure the system and at the same time, the user is alerted in several cases. The client is available for Mobile Android and desktop systems that support running applications in Java.

User, that is using this client, logs in to your account and is able to configure the different monitoring, show statistics, charts, graphs, etc. As the client runs at least in the background, the user is informed at the same time a form of notification about individual outages or other excessed monitored values. A big advantage for the client is just in its mobile version, where the user can be notified at any time for excessing the monitoring values, even with the minimum of data traffic.

This part of the system is designed as a thin client. This means that for its operation, we need some more applications. The client itself performs only a minimum of computational tasks, and most of the more complex operations are delegated to a different part of the system.

The client is implemented in both mobile and desktop versions, based on the event handler. Events will arise primarily from the user and its interaction with the user interface, and secondarily on the basis of the platform of asynchronous messages from the server part of the system (e.g. different types of warnings). For example, if the user wants to display all of its monitoring, the handler sends an event request to the data server of the system. After that, the application is ready to serve the next event and the incoming message from the server with the required data is then displayed to the user by the client.

Server

The second component is the server part (a.k.a. backend) performing the monitoring. It maintains the configuration settings stores values of individual tests into the database. At the same time it notifies the active clients of the critical events and, of course, makes other set of warnings. It also provides information to clients about the current configuration and be able to change these settings. On request, the client provides the statistical data obtained from the database.

This part is designed for monitoring at certain intervals the target services. It means it gets the status and values of the services and stores the values into the database. If it comes from the client the request of some data, the query is processed as a priority and the data is sent to client. In case of exceeding critical values set in the configuration of the monitoring, all active clients logged on to the account are notified, and there will be set warnings for the service.

In the standard running configuration, that the server part and the database run on the same host, but if there were problems with the lack of power of one server in larger amount of queries, or user for any reason, wish to have these two components separated, there is no problem to run the database on a separate server.

The key element of the implementation of the whole system is the backend testing core. Each monitoring is executed in a separate thread, where the monitoring is performed in a regular interval set by the user. The obtained data is then saved to the database.

An important characteristic is the ability to add custom modules, offering more types of monitored servers and services. This ability is a huge benefit, because the monitoring system is not dependent on a single set of functions, and thanks to its modularity we are able to adapt to the new requirements.

Agent

The last part of the monitoring system is the agent situated on the monitored computer. Its use is not mandatory and it is used for collecting system and performance information such as CPU load, the remaining capacity of the hard disk or system memory. The Agent runs in the background and listens on the specified network port. If it receives a query for specific system information, it finds the required information, sends it back and again switches to the listening mode. For

communication with the server part of the system, the agent uses its own communication protocol, which is similar to the protocol used for communication with the client.

Communication Protocol

For communication between the client and server parts, but also for the acquisition of data from agents, we proposed a simple proprietary protocol designed specifically for these purposes. The main emphasis was placed on its design simplicity, ease of parsing, low overhead and the intuitiveness of its understanding. Each message starts with one word (generally a string of characters) consisting of the uppercase letters of the English alphabet, which uniquely determines the type of the message. Then the data follow. The format of this data is given by the specific type of the message. The end of the message is identified by a character line breaks (\n). [1]

Conclusions

This paper proposed the system for monitoring of network servers, services and system resources. The system consists of the server to ensure the monitoring, service clients and work with the database. The client applications for desktop and mobile Android system are serving as the user interface for the entire system and gathering information from the agent deployed to the system resources of monitored servers. For communication between each part of the system, the proprietary communication protocol has been implemented.

Although there are many different applications for monitoring servers, the proposed solution is a unique combination of server monitoring and mobile client which offers truly continuous supervision of the servers and services. For comfortable configuration and monitoring, it can be used the desktop client application developed for standard PCs. Additionally, the system is extensible with plugins that may add other types of monitoring and allow the user to customize the system to his own needs. These features make the system created a unique tool not only for network administrators.

Acknowledgements. This project has been carried out with a financial support from the Czech Republic through the project no. MSM0021630528: Security-Oriented Research in Information Technology and by the project no. ED1.1.00/02.0070: The IT4Innovations Centre of Excellence; the part of the research has been also supported by the Brno University of Technology, Faculty of Information Technology through the specific research grant no. FIT-S-14-2299: Research and application of advanced methods in ICT.

References

[1] R. Fielding, J. Gettys, J. Mogul, Hypertext Transfer Protocol HTTP/1.1. RFC 2616 (Draft Standard) [online], (1999) updated by RFCs 2817, 5785, 6266, 6585. URL <http://www.ietf.org/rfc/rfc2616.txt>.

[2] F. Halsall, Computer Networking and the Internet. Addison Wesley, 5th ed. (2005) ISBN 0-321-26358-8, p. 832.

[3] J. F. Kurose, K. W. Ross, Computer Networking: A Top-Down Approach. Pearson, 6th ed. (2012) ISBN 978-0132856201, p. 864.

[4] J. Postel, Internet Control Message Protocol. RFC 792 (Standard) [online] (1981) updated by RFCs 950, 4884, 6633. URL <http://www.ietf.org/rfc/rfc792.txt>.