

On recording and presentation of measurement data acquired via web services

S. Miličić, I. Voras, H. Keko

Faculty of electrical engineering and computing

University of Zagreb

Unska 3, Zagreb, Croatia

E-mail: sonja.milicic@fer.hr, ivan.voras@fer.hr, hrvoje.keko@fer.hr

Web services have greatly eased exchange of data, but the usage of web services carries its own specific downsides as well as advantages. Most notable of these are latency of transmission and reliability of server equipment. In this work a straightforward solution for distribution and representation of measurement data was created. This solution is comprised of two parts: a back end process which communicates with the data source via a web service protocol based on XML (in Python language), and a front-end module for the FERweb Content management system (in PHP language). These processes communicate asynchronously via the database.

I. INTRODUCTION

This work presents a system whose purpose is to collect and present data from a remote measurement station via web services technology, as well as experiences gathered in the course of creating such a system. In this work the name “measurement station” will refer to an arbitrary complex system that includes sensors providing the data, as well as electrical and computer equipment that ultimately translate the data gathered by the sensors and expose them via the web services interface. The details of such system which was created and used for this work is out of scope of this paper and will be presented separately.

The web services are accessed by the FERweb Content management system (CMS) and the data displayed inside the standard web framework, as a potentially useful information provided to the web page visitors.

II. ON WEB SERVICES

In its strict definition, the name “web services” covers relatively large number of technologies that can be used for communication with remote computer systems that have the following points in common:

- Use of standard TCP/IP [1] networking infrastructure.
- Use of standard HTTP [2] communications protocol, on which the world wide web system is based, as the base transport protocol envelope for a higher-level protocol.

Major examples of web service protocols are SOAP (formerly an acronym for *Simple Object Access Protocol*) [3], XML-RPC (*Extensible mark-up language – remote procedure call*) [4] and REST (*Representational State*

Transfer) [5]. Of these, the first two use remote procedure call semantics for data access, i.e. the remote system exposes a number of procedures that the client can call with predefined arguments formatted according to a strict specification and receives a reply in a form of procedure return value. In both protocols XML is used to encode protocol-specific information. The REST is an architectural style for distributed systems and a canonical implementation using the HTTP that emphasizes extensibility and scalability rather than a uniform interface.

Since the inception and popularisation of web services, SOAP has become the most popular protocol, to the point that it is sometimes implicitly referenced as “the web services protocol.” Though it has been found to have interoperability problems because of its complexity (despite its former name, SOAP has become a complex set of specifications) it is now the dominant protocol with many commercial and free implementations, and as such is used in this work.

III. INFRASTRUCTURE

A. HARDWARE PLATFORMS AND NETWORK INFRASTRUCTURE

The measurement system and CMS system are physically located in different locations; the former is located in the *Department of power systems* and the latter in *Centre for IT support*. Both locations are a part of the same campus-wide network, but are situated in different buildings. Network infrastructure connecting the two systems is a standard consumer-grade implementation, with a 1Gbit/s backbone and 100Mbit/s switched connection inside the department and the centre. As such, this network is not optimised in any way for real-time communication or reliability, and both issues have had to be dealt with during the course of implementation.

Measurement station computer (which receives measurement data from sensors and exposes them as web services) is a custom built workstation-class computer based on 3GHz Pentium 4 processor, which also handles variety of departmental computing tasks. The CMS system runs on a server-class computer, Compaq (now HP) model ML330 with 1.3GHz Pentium III processor which hosts both the web server and the database, as well as several supporting services (such as mail server and system monitoring software). Both are standard COTS components in their class, similar to those used in most such environments.

Simplified path of measurement data is shown in Fig. 1.

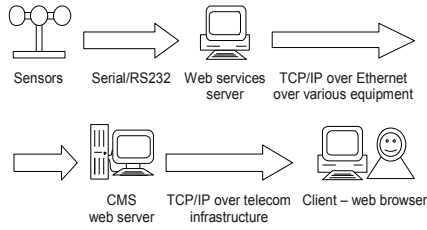


Figure 1. Simplified path of measurement data from sensors to the end user

B. SOFTWARE PLATFORMS

Web services are implemented using standard client-server methodology. In it, the system being used as the service provider (the measurement station) has a standard HTTP web server with web services extensions enabled. In this case, the server is a standard desktop-class computer system with Microsoft Windows XP operating system and Microsoft Internet Information Server (IIS) web server. The dynamic parts of web server, including support for web services, are provided by ASP.Net framework.

The client for measurement data in this work is another web server system, running the FERweb Content Management System, written in PHP language and using Apache web server on Linux operating system. In this configuration, web services protocol is effectively being used as a back-end protocol between the two server systems.

A. THE FERWEB CONTENT MANAGEMENT SYSTEM

The name “FERweb” is a colloquial name for the custom-made web Content management system (CMS) developed at the University of Zagreb, Faculty of electrical engineering and computing. The system is the result of over 4 years of development and has been successfully deployed as the Faculty’s web site, where it is used for collaboration and coordination between students and the academic staff. It is also deployed as the public web site of Croatian Academic and Research Network (CARNet) and was also sold to Pliva d.d. (a leading Croatian pharmaceutical company) where it is undergoing extensive development and customization as a part of a Knowledge management system.

FERweb is implemented entirely using Open-source software components, most important of which are PHP web programming language, PostgreSQL database and Apache web server software. One of its major strengths is an architectural emphasis on modularity of components. Each type of content and each functionality provided by the system is implemented as a “module” which is mostly self-sufficient and communicates with the rest of the system via a set of well-defined interfaces. This modularity allows for fast and efficient adding of additional functionality, and it is with a such a module that the measurement data is presented to the user.

IV. COLLECTING DATA IN THE CMS

The measurement station web services expose data from three sensors: temperature sensor, wind speed and wind direction.

First implementation of the CMS module was written in a way that it would contact the web services server each time the web page in which the module is located was refreshed. Due to pervasive use of content caching in the CMS, this did not mean a web service request for every single visit to the page, but the overall effect was still a noticeable slowdown for many users. Also, a problem was noted on the web services server where multiple simultaneous accesses to the server caused uncontrollably large delays because of competition for network resources and sensor hardware access.

Two solutions to this problems were considered. First solution was to use more aggressive caching/buffering of data on the CMS site or in the module, where on the first web page request to the CMS the module would collect data from web services and write them in the CMS database along with the time the request was made, so that the data could be marked as “expired” after a few seconds and another set collected when the web page gets loaded again. The second solution was a refinement of the previous one, in that the task of collecting measurement data was taken away from the CMS module and put into a separate operating system process. This process would contact the web services server to collect the data in fixed intervals (5 seconds) and write them to the CMS database. The CMS module would only need to read data from the database (a common and well-supported operation in the CMS) and format it for presentation to the user.

As the idea behind this work was to display approximate data to the visitors of the web page and not real-time data, the delay introduced by caching/buffering was considered to be allowable.

Each of the mentioned solutions was considered “good enough” in that the goal of reducing the frequency of contacting the measurement station was achieved, but the latter was considered to have an advantage in that there is no slowdown for any visit to the page (contention for data is handled in the database). The downside of the approach is that the frequency of contacting the web services server is constant, rather than optimistic (in the sense that there are no remote web services requests when there are no web page visitors) as is in the former approach. The “collector” process was implemented in Python language because it allows for easy prototyping.

Although the system was not meant to provide real-time data, a benchmark was made to determine the maximum performance that could be achieved using our particular setup.

TABLE I
RESULTS OF PERFORMANCE BENCHMARKS OF THE
WEB SERVICES PARTS OF THE INFRASTRUCTURE

Benchmark type	Results
Round-trip network “echo” packet time, 32 bytes payload	0.2 ms
Round-trip network “echo” packet time, 1024 bytes payload	1.2 ms
Web service request and reply, 690 bytes requests, 605 bytes replies	348 ms

Each benchmark in Tbl. 1 was made 10 times and the average value presented in the table. Benchmarks were conducted when the entire network was almost idle (in very late hours of the night). For each result, standard deviation was negligible and thus not reported.

From the results it is clear the the overhead of web services protocol and software infrastructure involved in transforming sensor data into the web services protocol is very high. Though the SOAP protocol, because of its use of XML and standard TCP/IP infrastructure requires significant amounts CPU power, the multi-layered architecture of the measurement station software is also causing significant delays in processing.

V. DATA PRESENTATION

Collected measurement data is displayed by the module in the CMS. The module makes use of standard CMS facilities for separation of data and presentation, and uses its own HTML template file for data presentation. Temperature and wind speed data are presented as ordinary numbers, but wind direction is represented as an indicator in a compass rose. A screenshot of the module is presented in Fig. 2. The module is usually located on the front page of the Department of power systems, <http://www.fer.hr/zvne>.



Figure 2. Web CMS module output

In case the viewer spends too much time looking at the web page, the module issues a web page refresh request to collect and display recent data.

During the course of implementing the system described in this paper, it became clear that a more general data collection system could be made, one which allows collection and tracking of many different types of sensors in a uniform way. Creation of such a system, which would offer generic graphical data presentation templates for

broad range of sensor types is planned in the foreseeable future.

VI. CONCLUSIONS

Web services offer an simple way of transferring data between the measurement station and the client (in our case, a web Content management system) unprecedented by previous technologies. With high-level software interfaces such as those offered by the SOAP protocol and those present in the CMS, building simple measurement data presentation facilities on a web page has become easier than ever before. Since the interoperability issues of web services protocols is either pre-existent or already solved, it was possible to implement a solution using very diverse infrastructure: COTS hardware of different purpose and quality which is located in separate locations, two different operating systems (Microsoft Windows XP and Linux) and three programming languages (C#, PHP and Python). But with the simplicity of implementation comes a price in performance. The concept of web services requires the presence of relatively complex network facilities and protocol handling software which is not present in embedded environments such as the sensors used in this work. Adequate speed of processing was only achieved by using a separate network-connected computer with enough processing resources to transform sensor readings into web services, and client-side caching in the CMS module.

REFERENCES

- [1] J. Postel, ed., *Transmission Control Protocol*, Internet Engineering Task Force (IETF) Request for Comment (RFC) document #793, 1981.
- [2] R. Fielding, J. Gettys, J. Mogul et al, *Hypertext Transfer Protocol – HTTP/1.1*, IETF RFC document #2616, 1999.
- [3] M. Gudgin, M. Hadley, N. Mendelsohn et al, editors, *SOAP Version 1.2*, W3C Consortium Recommendation, 2003.
- [4] D. Winer, *XML-RPC Specification*, <http://www.xmlrpc.com/spec>
- [5] R. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, Ph.D. Thesis, ch. 5, University of California, 2000.
- [6] P. Prescod: *Roots of the REST/SOAP Debate*, online publication, http://www.prescod.net/rest/rest_vs_soap_overview/