웹 기반의 요소관리와 네트워크관리를 위한 내장형 웹 서버 구조

Embedded Web Server Architecture for Web-based Element and Network Management

DECE 19973096 주홍택, Hong-Taek Ju, Embedded Web Server Architecture For Web-based Element And Network Management, 웹 기반의 요소관리와 네트워크관리를 위한 내장형 웹 서버 구조, Department of Computer Science and Engineering, 2001, 90P, Advisor: James Won-Ki Hong, Text in English

ABSTRACT

Our research work focuses on Web technology for building a scalable and efficient network management environment. Web technology has been employed in network management for a number of years and nowadays it is accepted as a proven technology in network management area. However the usage of the technology was limited in areas of accessing and presenting information. Web technology is useful not only for simple and effective user interface but also for building key components in network management applications, such as data processing, analyzing and distribution. In the light of recent advancement of Web technology, there are many available opportunities for solving most of management problems of evolving network, comparing the capabilities of the traditional management framework with Web-based network management.

Web-based element management gives an administrator the ability to configure and monitor network devices over the Internet using a Web browser. The most direct way to accomplish this is to embed a Web server (EWS) into a network device and use that server to provide a Web-based management user interface. In this dissertation, we present the design and implementation of a Web-based element management architecture. The architecture provides basic interface mechanisms for use between a management application of an embedded system and an EWS. An integration mechanism between a Web document and a management application enables developers to add new management functionalities by merging Web documents with management application programs that generate dynamic management information. For rapid and low cost

development, an easy but powerful integration mechanism must be provided. We suggest effective integration mechanisms for each interface mechanisms. We validate the effectiveness of Web-based element architecture by developing Web-based management user interface for commercial Internet router.

We then extend the Web-based element management architecture to support the Web-based network management, which provides a network-wide view of management information: managing the state of network link, topology and policies as well as the network elements. As to extending the architecture, we apply XML technology to the management information exchange between management applications, and management information processing performed by the key components of the architecture. XML is a new standard data-exchange format over the Web. As the XML related technologies are added, it can support not only for data exchange but also for data processing such as parsing, translation and logging. The main advantages of using it are easy to use, simple but powerful, large support by application software, openness and inexpensiveness. But these advantages cannot be maintained in network management application without appropriate methods for applying XML to network management. By taking advantages of XML technology, the Web-based network management architecture gives a way to develop management applications efficiently and to manage devices effectively. We validate the effectiveness of the Web-based network management architecture by developing a Web-based management system for a commercial ultra-dense Linux server based on the architecture.

Contents

List of Figures

List of Tables

1. Introduction

This chapter gives an overview of the network management and Web-based management. It highlights explanation on Web-based element and network management.

1.1 Network Management

Network management is the sum of all activities related to configure, control and monitor network and systems. The goal of these activities is to ensure the effective and efficient operation of systems and communication network. The evolution of network management has been in close systems and communion network with the way in which systems and network themselves have evolved - from a limited interconnected homogenous set of systems under one domain to a large heterogeneous distributed communication environment spanning across multiple domains. It is evident that the complexity of network management has accumulated over the years to cater to the heterogeneous and ubiquitous communication networks that we have today. The complexity has been a direct consequence of variety of network components, geographic distribution of components, multiple operating domains, integrated service environment and heterogeneity of systems.

The standard specifications that have a cross-system and multi-vendor orientation are the prerequisites to solve heterogeneity and distribution of managed resources. A framework consists of all the necessary specifications and management architecture. Along with the standard specifications, management architecture provides the basis for different manufacturers to develop interoperable management applications largely independent of one another.

Generally it introduces the four models: information, organization, communication and function.

The management architecture that most successfully incorporates these four models from the conceptual standpoint is OSI management architecture of International Standard Organization (ISO) [1] and International Telecommunication Union (ITU) Recommendation M.3000 [2], which also represent the basis for the telecommunications management network (TMN). The Common Management Information Service/Protocol (CMIS/P) [3,4] has been standardized by ISO and recommended by ITU. CMIP implies the use of a full protocol stack according to the OSI Reference Model [5]. The Manager/Agent model of distributed management processes originated from this OSI work. The tools available for system developments involving the use of CMIP have increased in quality and availability. Some developers still consider it a cumbersome and expensive technology. It is considered to have major strengths when it comes to managing network elements of moderate to high complexity but too heavy for the relatively simple network elements in data networks.

The IETF sought a means to control and monitor devices attached to the Internet and came up with the Simple Network Management Protocol (SNMP) [6]. SNMP was designed in the late-1980s to provide a simple means of effective management on different types of networks. Its initial aim was to be an interim solution until a better designed and more complete network management scheme became available. However, for many people no better choice became available and SNMP became the network management protocol of choice for IP-based data networks. Although SNMP itself is relatively simple, it takes some work to build agents and considerably a great deal of work to build and deploy managers onto the various network management platforms.

1.2 Web-based Management

Web technology is very rapidly penetrating many business areas. Systems and network management is no exception. The technology is based on Internet and offers a number of benefits in terms of openness and ubiquity of its standards and tools. The ability to use a universal browser to access management functions, device status and statistics, and to configure remote managed objects from anywhere at anytime give many advantages to a network administrator. For developers, system development cost and time can be saved by standardizing on browser instead of workstations, and by use of existing standards and numerous supporting tools. Recently, two approaches have been proposed for Web-based management by industrial standardization bodies: Web-Based Enterprise Management (WBEM) [7] and Java Management eXtension (JMX) [8]. The result from WBEM is fairly stable, but still not quite ready to deploy. JMX's technology dependency on Java [40] results in less generality, especially for embedded environment. Web-based technology for network management has already made the breakthrough with different practical solutions. Those solutions are fragmented and concentrated on viewing and information distribution capabilities. Researchers have extended the HTTP/HTML [9,10] by Java [11], Web push, XML [12], dynamic Web technologies, and etc. Although these technologies are rapidly maturing, the developer of Web-based management system is pretty much left to his own idea to figure out how to link the technologies to the management system development. No guidelines exist to help put all of these technologies and standards into perspective.

1.3 Web-based Element Management

Most simple and typical case in applying Web technology to network management is to embed the Web server into a network device for element

management. This type of Web server is called an Embedded Web Server (EWS) [13,14,15]. The EWS provides a Web-based management user interface constructed using HTML, graphics and other features common to Web browsers. Figure 1 shows the Web-based Element Architecture.

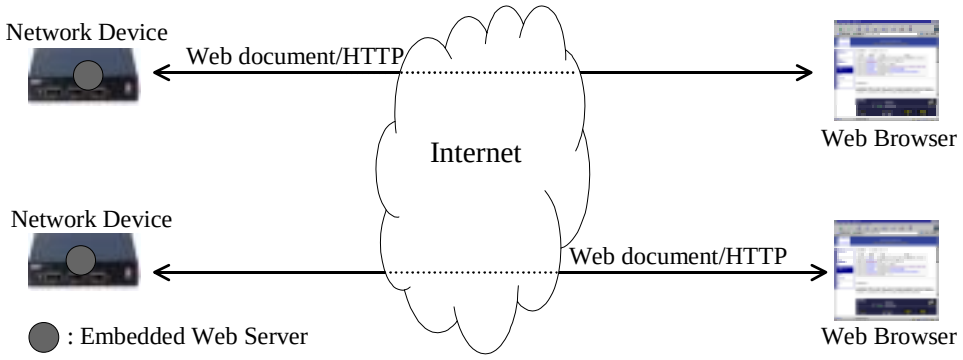


Figure 1. Web-based Element Management Architecture

The status of device is provided to the user by simply retrieving pages, and operator command is sent back to the device using forms that the user completes. Web-based management user interfaces through embedded Web servers have many advantages: ubiquity, user-friendliness, low development cost and high maintainability.

EWSs have different requirements, such as low resource utility, high reliability, security and portability, for which general Web server technologies are unsuitable. Above all, due to resource scarcity in embedded systems it is important to make EWSs efficient and lightweight. There are also design issues such as HTTP [10,16] and embedded application interface.

1.4 Web-based Network Management

Web-based element management has its limitations with respect to scalability – configuring hundreds of routers and switches via a Web browser is

simply not scalable. If there are many EWS-equipped network elements that is typical for large networks, an administrator must open a Web browser for each device. This scenario ignores the realities for large networks. This approach also tends to be device-centric and may not be able to provide logging, network topology, trend analysis and other high-level management capabilities that are normally essential for network management.

In Web-based network management, the manager runs as an application over the operating system, and collects and disseminates the information gathered from the network and systems device to the browser. In doing so, the manager needs a standard access protocol in order to support a multi-vendor environment. HTTP [10,16] is used as a management protocol between the manager and agent because Embedded Web server already provides it as an access mechanism. It is also necessary to devise a format for data exchange between two programs over HTTP. The W3C responded to this need for a data oriented interchange format by defining the eXtensible Markup Language (XML) [12]. Figure 2 shows Web-based network management architecture.

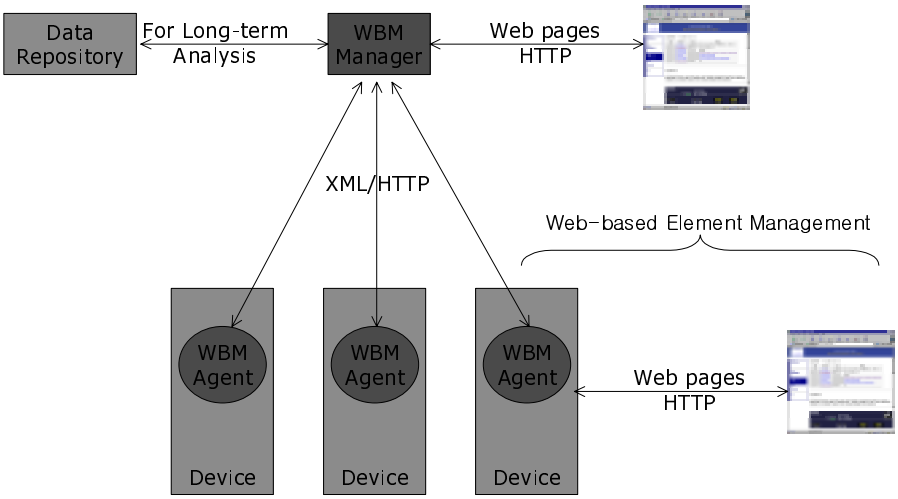


Figure 2. Web-based Network Management Architecture

Web-Based Management agent (WBM agent) and Web-Based Management manager (WBM manager) perform agent and manager tasks, respectively. The WBM agent and WBM manager exchange management information over XML/HTTP. The WBM manager collects management information from multiple WBM agents and stores them in the information repository for long-term analysis. After all, the WBM manager is XML-based Web client application for network management. And the WBM agent is a specialized Web server for providing management information in XML form.

There are many potential advantages of using XML in network management: fully extensible by user-defined vocabulary, self-describing, structured information and manipulated by standard mechanism. However there remain a number of questions pending before the potential of XML can be exploited. Specifically, we need to delve into the following issues:

- The ability to meaningfully mark up information, such that it can be manipulated, exchanged and stored mechanically.
- A generic information model adapted to the representation of network equipment and data.
- Means to dynamically update the information in the model and notify the administrator of relevant changes as the network evolves.

1.5 Problem Statement

In this chapter, we describe the problems to be solved in the development of Web-based management systems. Firstly, we arrange the issues in developing an embedded Web server in order to use it for network management. When someone develops a Web-based element management system using the embedded Web server, his question is how to develop the system efficiently and effectively. Secondly, to answer the question, we describe things to be provided by Web-based management architecture to the developer. Finally, we examine what problems to

consider so as to extend Web-based element management to Web-based network management.

1.5.1 Embedded Web Server

A Web server can be embedded in a device to provide remote access to the device from a Web browser if the resource requirements of the Web server are reduced. The end result of this reduction is typically a portable set of code that can run on embedded systems with limited computing resources. The embedded system can be utilized to serve the embedded Web documents, including static and dynamic information about embedded systems, to Web browsers.

- ***Resource Scarcity*:** The development of EWS must take into account the relative scarcity of computing resources. EWS must meet the device's memory requirements and limited processing power. General-purpose Web servers have evolved toward a multi-threaded architecture that either dedicates a separate thread to each incoming connection, or uses a thread pool to handle a set of connections with a smaller number of threads. Single process or thread to every incoming connection is usually impractical due to the memory overhead required and, in some cases, to the lack of system support for multiple processes.

  More difficult than memory limitation is managing the impact of Web request servicing on the system CPU: Can request processing be done in a way that allows the rest of the system to meet its real-time constraints? An EWS process as a subordinate process to the main purpose of the device must use as little CPU as possible, in order not to interfere the main task of system. To minimize system resource usage, EWS can place restrictions on some parameters. For example, it is not necessary to support a very large number of connected users; usually one to a half dozen users will be accessing the system at a time.
- ***Reliability and Portability*:** Generally network devices require high reliability. As one embedded component of network device, EWS also must be highly

reliable. Because it is a subordinate process, at least it must protect propagation of internal failure to the whole system. EWS needs to run on a much broader range of embedded system in RTOS environments that vary widely in terms of the facilities they provide, and with much tighter resource constraints than mainstream computing hardware. So it requires high portability.

- ***Security*:** EWS must provide a mechanism to limit access to sensitive information or configuration control. EWS should look to incorporate Secure Socket Layer (SSL) [18] protocol, which ensure a secure socket connection between a browser and Web server. There is also Secure-HTTP, an extension of HTTP for authentication and data encryption between a Web server and a Web browser [42]. Even public and private key technologies can be leveraged to control access to managed device. There are different levels of security in Web environment from no security to simple, medium and strong security. EWS developer must take into account for what security level is moderate to the Web-based element management system.

1.5.2 Web-based Element Management

In order to provide Web-based element management, developers put Web pages into the device after embedding Web server into it. Web pages must be stored in the embedded system to be served at run time. A developer can choose an appropriate interface mechanism depending on the management information characteristics or types of Web documents. An integration mechanism between a Web document and a management application, source of managed resource status, enables developers to add new management functionalities by merging Web documents with management application programs that generate specific dynamic management information. For rapid and low cost development, an easy but powerful integration mechanism must be provided. Also effective integration mechanisms for each interface mechanism separate Web document development

from management application development.

- ***Interface Mechanism*:** While Web documents can be quickly prototyped with readily available desktop authoring tools, that prototype must then be integrated into the system software. EWS works with a fixed set of integrated Web documents that are usually frozen at the time that the embedded system is manufactured, but incorporates dynamic information of system software at run-time

  Management information can be classified on the basis of update period, direction of information flow or object of information source. From the viewpoint of update period, some management information changes dynamically, and some does not change at all. Further, some information possesses real-time characteristics, for example a dynamic graphic form in a real-time update or event notification to perform some action. Regarding the direction of information flow, some information can originate from a Web browser and travel to a Web server and vice versa.

  It is desirable to provide several effective interface mechanisms between the Web document and management applications. This variety in determining the interface mechanism enables developers to choose an appropriate mechanism on the basis of characteristics of management information and Web documents.

- ***Integration Mechanism*:** An integration mechanism between Web documents and management applications enables developers to add new management functionality by merging Web documents, which provides a user interface format with management application programs that generate specific management information. Through the integration mechanism, Web document development can be separated from management application development. Consequently, the management application developer no longer needs to consider the user interface of Web documents through program logic.

  Also, Web document prototyping, involving different user

interfaces with a deployed Web document, no longer requires code changes and recompilation of the management application program if the same management application interfaces are used and if a Web page can be uploaded dynamically. For rapid development, an easy but powerful integration mechanism must be provided.

1.5.3 Web-based Network Management

In case of Web-based element management, EWS represents actual status information of systems and network on a Web page to be accessed through Web browser. On the other hand, Web-based network management regards a Web client application program, not Web browser, as communication partner. The Web client program collects, stores and analyzes the management information retrieved from the network devices via HTTP connections. Eventually Web-based network management diverts the usage of HTTP from a general Web document delivery to management information exchange.

- ***Dual Management Stacks*:** The Simple Network Management Protocol (SNMP) [6] has provided the only widespread network management solution in the industry for the past two decades by providing interoperability framework to collect instrumentation level measurement data to support fault and performance management. Web-based element management improves significantly configuration management, remote diagnosis and remote troubleshooting. Each management scheme has mutually exclusive functionality and advantages; consequently two management schemes coexist in real world.

  The combination of both technologies increases the requirement of computing resources in network devices due to supporting dual management stacks. If the same managed device is supporting two agents, one for SNMP [6] and the other for HTTP [10,16], the overhead is too high. Also, it is hard to guarantee for set synchronization, consistency and access control because there are multiple access paths to the managed real

resource. In order to avoid above problems, mapping from SNMP to HTML/HTTP within a device can be used. That is more than just perusing MIB variables with a Web browser. In our architecture, we unify management protocols into HTTP.

- ***No Centralized Configuration Management*:** As mentioned before, Web-based element management allows operators to access management information through Web browser. Since embedded Web server normally provides management information in a form of HTML/Java that is a natural language or a program with display logic, an operator can understand the meaning of management information, by looking at display of the browser. But it is almost impossible to let computer software understand it by interpretation of HTML/Java. Therefore configuration management information disseminated by embedded Web server cannot be collected or processed by Web-based manger application based on the manager-agent paradigm.

  Due to the absence of centralized management capability, management functions are limited: impossible for data preservation, aggregation and report generation. Because Web-based element management that is strongest in configuration management gives only browser access, there is no centralized way for configuration management and remote diagnosis In this thesis, we present a method that makes it possible to build centralized management applications based on the manager-agent paradigm.

- ***Use of SNMP*:** SNMP is still in use for the case of inappropriate due to its low scalability, inefficiency and unbalanced manger-agent role. SNMP is essentially implemented in all network elements. Due to its widespread implementation, it is not going to go away anytime soon. However in our opinion, it will quickly become relegated to legacy status. SNMP will no longer be the center of innovation and progress in management standardization. There are many examples of legacy systems remaining

important and widely used for decades after losing their status as leading edge, and SNMP will be one of them.

SNMP is not powerful enough to provide what is needed for managing ever-evolving complex, large and dynamic network environment. In this paper, we do not make a long-winded explanation about SNMP weakness, but make a list based on the previous work done by other researchers in Section 3. The study of SNMP weakness guides us to design more advanced architecture than SNMP. Our architecture can be alternative to SNMP where the efficiency and effectiveness is more important than interoperability. We also answer the question of how to use legacy SNMP agents in the proposed embedded Web server architecture.

WBEM can be relevant solution to the above problems. But we identify two problems in WBEM: heavyweight and excessive domain specific solution. Detailed analysis for these problems in WBEM is described in section 2.2.1.

1.6 Research Approach

In the first stage of our research, we developed an efficient and lightweight embedded Web server for Web-based management. Developed embedded Web server supports all essential Web server functions to be used for management application in embedded systems. These functions include persistent connection, cache control, cookie mechanism, security and application interface mechanisms. In order to save computing resource at runtime, we developed HTTP engine of the server as an finite state machine and introduce various optimization technique such as compression and preprocessing. High portability and reliability can be achieve by reduce dependency on other component such as operating system and system libraries. We eliminate code dependency of developed embedded Web server by incorporating necessary functions into Web server code. We added

security module that accomplish user authentication by checking username/password.

We designed an Web-based element management architecture. The architecture was designed to be a base of implementation for Web-based element management user interface that support all necessary functions with minimal developer's efforts. We defined the interface between Web documents and management application of embedded system. By defining the interface, we can separate the display and production logic of management information. Web-based element management architecture has four interface mechanisms with their own unique advantages. Effective integration mechanisms were introduced for interface mechanisms. We demonstrated effective use of these interface and integration mechanisms by implementing Web-based element management system for a commercial Internet router.

We extended the Web-based element management architecture to support Web-based network management architecture. We adopt XML as an enabling technology in providing solution of the extension. The capability of XML schema model is used for management information modeling and the encoded XML document of management information is exchanged in manger-agent communication. XPath is used for addressing mechanism over the managed object. Also we apply XML DOM to uniform representation and unified the access mechanism of management data in a WBM agent. By the appropriate and effective use of XML technologies, we have maximized the advantages of using XML in network management.

1.7 Organization of Thesis

The organization of this dissertation is as follow. Chapter 2 is devoted to Web basics and presents the current state of Web-based network management. Chapter 3 and 4 focuses on the solution for Web-based element management and

Web-based network management, respectively. Chapter 5 discusses the implementation and validation of proposed solution. Finally, we conclude and give directions for future work in Chapter 6.

2. Related Work

This chapter offers discussions about the applicability of Web technologies for network management and addresses the three principle standardization efforts: IETF [17], JMX [8] and WBEM [7]. Also previous research work from academia and institution are discussed.

2.1 Applicability of Web Technologies

In this section, we discuss the key Web technologies from the viewpoint of Web-based management.

2.1.1 HTTP

In May of 1996, the IETF published an informational memo (RFC 1945) to document the HTTP/1.0 protocol [16] as it was then being used, but there were sufficient problems with that protocol that it was not formally a part of the Internet standards. Since that time, the HTTP/1.1 protocol [10] has been developed to address those problems. The major improvements made in HTTP/1.1 are both important to embedded system interface developers: explicit cache control and persistent TCP connection.

The improvement of explicit cache control in HTTP/1.1 are designed to allow appropriate caching of responses, either by private caches within a browser or by intermediate systems such as proxy servers or cache severs. Both clients and servers can control when it is appropriate to cache a response or to use a cached response. For Web documents that do not change (such as logos and other embedded graphics, or pages containing help text), this is a great performance win because at worst a very small request and response are needed to verify that the

cached copy is still valid. The mechanisms for this in HTTP/1.0 were not sufficient, and what was included uses timestamps, presenting a problem for embedded system implementations that may not have a synchronized (or even any) clock. HTTP/1.1 caching can be controlled without any clock. Caching is desirable for static Web documents, but dynamically generated Web documents must not be cached in order to retrieve up-to-date management information.

HTTP/1.0 uses each TCP connection for just one exchange - the browser sends a single request and the server sends the response and then closes the connection. This creates both resource and performance problems because a number of requests are required for typical Web pages (one for main page, one for each frame, and another for each embedded image). Because each TCP connection requires resources for buffering and maintaining the connection state, the memory requirements for the system are increased. Moreover, TCP implementations maintain connection state information for two minutes after connection closed. Performance is reduced because of the extra round trip delays required to set up each connection, and because the TCP congestion avoidance algorithms cause each new connection to artificially restrict throughput, increasing it only gradually to discover the maximum available bandwidth.

HTTP/1.1 also includes improvements in connection usage to allow multiple requests to be pipelined on a single connection, with a mechanism to encode each response as a series of chunks so that it is not necessary to buffer the entire response before transmitting it. The result is better performance for the user at lower resource cost for the embedded system. It remains important that the embedded system be able to handle multiple simultaneous requests; because most HTTP/1.1 browsers open two connections for a typical page, splitting the requests for the page content between them, and because there may be concurrent requests from different users.

2.1.2 Web Documents

Web browsers have become widely popular because they all share

understanding of a simple media type: HTML [9] formatting language. HTML is optimized for display rather than printing or storage: no notion of pages and data format. HTML documents can be interconnected by specifying links that allow the user to move from one document to another. Links are visually displayed on the Web pages as pictures or underline text. In network and systems management environment, operators, who monitor constantly the health of the networks, typically follow well-defined procedures to configure the systems and network and to identify the cause of a problem of them and correct it. Link in the Web pages guide the procedure. HTML pages are static, and dynamic updates are not really supported. Dynamic attribute is absolutely necessary when Web technology is to be successfully implemented for network and systems management. But now with dynamic HTML (DHTML) [45], new client-side technologies, combined with scripting languages like JavaScript, may solve many of the problems with HTML.

Embedded Web server software must provide mechanisms for the embedded application to generate and serve Web pages to the browser and to process HTML form data submitted by the browser. One possible solution is modeled after the Common Gateway Interface (CGI) [19] found in many traditional Web servers. In this model, each URL [41] is mapped to a CGI script that generates the Web page. In a typical embedded system, the script would actually be implemented by a function call to the embedded application. The application could then send raw HTML or other types of data to the browser by using an interface provided by the embedded Web server software.

Another solution is to use Server-Side Include (SSI) [9]. With this approach, Web pages are first developed and prototyped using conventional Web authoring tools and browsers. Next, proprietary markup tags that define server-side scripts are inserted into the Web pages. The marked-up Web pages are then stored in the device. When a marked-up Web page is served, the embedded Web server interprets and executes the script to interface with the embedded application. For example, a proprietary scripting language could define an

interface to invoke application functions used to generate the dynamic part of a Web page. Embedded system initialization code is used to register the server-side scripts with function calls. The code for a function call is invoked when the server interprets server-side scripts. Active Server Page (ASP) and Java Server Page (JSP) are widely applied standard method in the way of SSI [9].

2.1.3 Java Applet

HTTP & HTML only scheme is client driven. One of the side effects of this is that once a page is served to the Web browser it becomes static and does not change even if management data has changed on the server side. For a user looking at a dynamic device, this is not very useful. To be truly useful for management application, pages will have to be constructed dynamically so that real-time data can be placed along static HTML in the same page. For common types of real-time data such as traffic monitoring and CPU load, users want to see data displayed in a dynamic graphic form. This is where Java applet [11] comes in. Java applets are automatically downloaded by a browser as a separate application that gets used within an HTML page. Once the applet is loaded, it has control over where it gets its data and how to display or manipulate that data. Java applets by nature are cross-platform and will act the same within any browser.

Another situation we have to consider, where the network device needs to notify a user of an event and possibly allow the user to perform some action based on it. The event notification is another issue sine there is no asynchronous communication method in HTTP. A Web browser would have to refresh a page or the page would have to have client-pull refresh periodically to see if there is new event. It is straightforward to design an applet to handle this kind of thing. Normally the browser must be active with that applet at all times and new server for sending event notification or responding query for new event must be introduced into the network device.

HTML provides the Web document designer with a large number of options and already well suited to developing rich interface. But it is static,

constraining the interactivity due to the response time limits of the network connection, and does not support for some powerful user input method such as mouse tracking. Java applet can use abundant user input library that resides in browser running computer.

It is a commonly accepted that Java applet has a small footprint while the Java Virtual Machine (JVM) required to execute byte code of Java applet is typically very large. Because Java applet containing byte code only is fairly compact, it can be stored in embedded system without large increment of memory capacity. Since Java is not actually executed on the network device, a JVM is not required.

2.2 Standard Activities

There are two promising approaches in Web-based management from industrial standardization bodies: Web-Based Enterprise Management (WBEM) [7] and Java Management eXtension (JMX) [8]. WBEM multi-vendor alliance launched in July 1996 and worked for establishing standards for Web-based network management software. In 1997, WBEM adopted HTTP as its transfer protocol and selected the Extensible Markup Language (XML) [12] as a representation for management information.

2.2.1 WBEM

Web-Based Enterprise Management (WBEM) [7] is an industry initiative to develop a standardized, non-proprietary means for accessing and sharing management information in an enterprise network. WBEM provides this common model, namely the Common Information Model (CIM) [21,22]. CIM is an object-oriented schema of managed objects. These managed objects are representations of real resources, and the schema provides a single data description mechanism

for any data that they may provide. WBEM provides an information standard that defines how data is represented and a process standard that defines how the components interact.

Obviously, a method for accessing CIM-provided data is required. The DMTF defined CIM to XML mapping and CIM operations over HTTP [7]. The specification of the CIM to the XML mapping standard defines the XML schema used to describe the CIM meta-schema. Both CIM classes and instances must be valid XML documents for that schema. The CIM operation over HTTP allows implementations of CIM to operate in an open, standardized manner. It describes the CIM operations that form the HTTP payload using XML and the syntax and semantics of the operation requests and their corresponding responses. WBEM uses XML only for object and operation encoding. By accepting XML technology in more positive light, WBEM can maximize the advantages of Web technology.

In section 1.5.3, we define two problems in WBEM; heavy weight solution to embedded system and excessive reliance on domain specific technology. First deficiency of WBEM is that its solution is too complex to apply it to embedded system. Therefore it cannot be used for unifying access path and an alternative solution to SNMP. Second problem is that WBEM heavily depends on domain specific technology; new management information model and new management operations. This disadvantages result in a lack of supporting tools, high development cost.

2.2.2 JMX

Another promising approach to the Web-based management is being realized by Sun: JMX (formally JMAPI) [8]. Sun announced JMX in order to provide ubiquitous management framework and promote an abundance of management application in Java. Based on the early JMAPI work as well as research taken from Java DMK development, JMX released final version 1.0 [25] and is awaiting completion of the reference implementation.

The JMX specification defines the interface for basic services as a registry

(Mbean Server) for Mbeans (JavaBeans for management). These services enable agents to manage their own resources and let managers forward information back and forth between agents and management applications. In the JMX architecture, both services and devices are treated as managed objects. The components, Mbeans, can be added and removed as needs evolve. Appropriate protocol adapters can provide a recognizable object to the Browser or JMX manager whose specification is under way. JMX depends greatly on Java. In order to be instrumented in accordance with the JMX, a resource must be fully written in the Java programming language or just offer a Java technology-based wrapper. Java Virtual Machine is a basic requirement for the management application. This heavy technology dependency on Java results in less generality.

2.2.3 Comparisons

In Table 1, we compare WBEM, JMX and SNMP. They share basic architectural elements: agents or object manager, sometimes server. The elements are in charge of managed resources. The management information from several remote agents is collected and accumulated at a central location by a manager, using a suitable intermediary communication protocol. In WBEM world, there is no dominant terminology for these elements.

The information model of SNMP, known as SMI, is defined as tree structured simple variable type and based on object-based paradigm. WBEM and JMX use an object-oriented paradigm for information modeling. CIM, information model of WBEM, includes standard models from hardware and software elements to global policy and end-to-end state information. CIM is broader but still shallow than SNMP MIBs. CIM is based on the UML a design method for specifying, visualizing and documenting object-oriented systems. CIM also has text-based notation method (MOF) and automatic translation algorithm (CIM to XML mapping).

Features	SNMP	WBEM	JMX
Architecture	Manager-Agent	Clams to support all	Manager-Agent
Information Model	Object-based	Object-oriented	Object-oriented
Specification Language	SMI	CIM (MOF, UML, XML)	Java
Operations	Get, Set, Trap	23 operations	Not specified
Communication Mode	Sync/Async	Sync	Not specified
Addressing	MIT with OID	Name and Associations	Java object name
Standardization Body	IETF	DMTF	Java Community
Mgmt. Domain	Network Mgmt.	Systems Mgmt.	Unidentified
Protocol Suit	UDP	HTTP/TCP	Not specified

Table 1. Feature comparisons of SNMP, WBEM and JMX

JMX deals with management instrumentation of Java software or of other entities that receive Java-based wrapper. The management interface of the managed Java object is also defined in Java language. JMX is deliberately designed as model-agnostic and it imposes no particular protocol. There are facilities for introducing a rage of protocol adaptors as needed. WBEM is more complex and advanced than SNMP: object-oriented information model, lots of diverse operations, flexible naming scheme. WBEM support is gaining ground, particularly in system management.

2.3 XML-based Network Management

This section presents the state of the art in XML-based network management.

2.3.1 XML

The eXtensible Markup Language (XML) [12] is a World Wide Web Consortium (W3C) standard based upon SGML[46]. The XML makes it possible to define its own markup languages with emphasis on specific tasks, such as electronic commerce, supply-chain integration, data management, and publishing. For those reasons, XML is rapidly becoming the strategic instrument for defining corporate data across a number of application domains. The properties of XML markup make it suitable for representing data, concepts, and contexts in an open, platform-, vendor-, and language-neutral manner. XML schema describes the data model of XML document.

Whereas HTML is documents presentation oriented, XML allows the flexibility to define data. XML has taken the software industry by storm and its repercussions will continue to be felt for some time. XML promises to simplify and lower the cost of data exchange and publishing in a Web environment. It is a text-based syntax that is readable by both computer and humans and offers data portability and reusability across different platforms and devices. It is also flexible and extensible, allowing new tags to be added without breaking an existing document structure. Based on Unicode, XML provides global language support.

There are two fundamental approaches to defining XML schemas: Document Type Definition (DTD) [47] and XML Schemas [27]. Either approach is sufficient for most documents, yet each one offers unique benefits. DTD is in wide spread use and shares extensive tool support, while XML Schema are more powerful and provide advanced features such as open content models, namespace integration and rich data typing. As having high opinion on rich data typing, we

decided to use XML Schema approach for specifying management information.

Document Object Model (DOM) [29] specifies the means by which you can access and manipulate your XML document. Without DOM, XML is no more than a storage system for data that can be accessed by various proprietary methods. With DOM, XML begins to approach its promise as a truly universal, cross-platform, application-independent programming language. DOM allows reconstruction of the complete document from the input XML document, and it should allow access to any part of the document. Also it has a series of factory methods that allow manipulation, additions, and deletions to the original document. In relation to WBM agent, we use DOM as a uniform access to management data and a central storage area for management data.

XPath [48] is an expression language that is used to address parts of an XML document. The syntax used by XPath is designed for use in URIs [49] and XML attribute values, which requires it to be very concise. The name of XPath is based on the idea of using a path notation to address XML document. XPath operates under the assumption that a document has been parsed into a tree of nodes. We use XPath as the basis for addressing managed objects between the WBM manager and WBM agent

Beyond the general-purpose advantages of XML stated above, XML also has numerous advantages in the specific case of network and systems management. As XML is becoming ubiquitous, it offers wide skills base and the range of tools and system integration mechanism its broad applicability provides. And the expressive potential of XML make it possible to represent the wide range of information models present in existing management standards and deployed solutions. Further, the separation of information represented in XML form the mechanisms to present, transmit and store that information make it a flexible solution for the specification and manipulation of corporate management data.

Under the XML umbrella are several supporting standards, which include

- Rules for representing structured data: Resource Description Framework (RDF) [26].

- A way to specify vocabularies and schema: Document Type Definitions (DTD) and Schema Languages [27].
- A language for design information: XML Metadata Interchange (XMI) for encoding UML [28].
- A way to program with XML content: Document Object Model (DOM) API [29].
- A way to format data in browser: eXtensible Style Language (XSL) [30].
- A way to query and transform content: XML Query Language (XQL) and XSL Translation (XSLT) [31].

There is a good possibility that these technologies play an important role in the evolution of network and systems management.

2.3.2 XML for Management Communication

XML over HTTP has been used for the definition and encoding of messages in data communication protocol. As described section 3.3.1, WEBM, active user for XML, defines a set of XML definitions for messages that exchange information on managed objects classes and instances using XML/HTTP. J.P Martin Flatin suggested an integrated management architecture based on Web technology, called Web-based Integrated Management Architecture (WIMA) [32]. In the WIMA, XML is used for representing management data, which is mapped from well-known standard management objects such as SNMP MIB variables, CIM objects, etc. But he introduced only mapping methodologies, model-level and metamodel-level mapping, without concrete algorithms. He also suggested how XML can be used for high-level semantics and unified communication model. But its description is so superficial that it needs to be improved in more detailed version. In section 3.3.4, we describes the WIMA in detail.

John, et al. in 1995, suggested an example of XML-based management communication architecture, XNAMI [33]. In the XNAMI architecture, the XNAMI manager can transfer compressed Java bytecode so as to add SNMP MIB.

The agent maintains an explicit runtime representation of its SNMP MIBs and uses XML to represent managed object internally. The agent stores Java bytecode for the GET and SET methods. Upon receipt of an SNMP GET, the agent's SNMP server executes the GET method, which is added by XNAMI manager. With a Java bytecode, the XNAMI manager transfer an XML string specifying that a new OID should be created, an existing OID dele. The XNAMI manager and agent exchange management data via SNMP, while the XML data is transferred via HTTP. Eventually, XML/HTTP is used for configuring SNMP agent.

2.3.3 XML for Management Model

XML is a metalanguage -- a language for describing other languages -- that enables an application (such as network and system management) to define its own markup. XML allows the definition of a customized markup language specific to an application. In network management domain, it has strong potential in the specification of management information.

Recently, C. Ensel and A. Keller reported their research results for applying XML, Xpath and RDF to the problem of describing, querying and computing dependencies among services in a distributed computing environment [34]. The definition of an XML based notation for specifying dependencies facilitates information sharing between the components involved in service process.

2.3.4 Web-based Integrated Management Architecture

The most sophisticated and comprehensive research work for Web-based management is Web-based Integrated Management Architecture (WIMA) [32] by J. P. Martin-Flatin. In his Ph.D thesis [32], he identified problems in SNMP-based management very well. Below list is a quotation from his thesis.

- Scalability and efficiency issues: difficulties in ever more management data, too many messages for bulk transfer, latency in sparse table retrieval, verbose OID naming, no compression of management data,

polling, unreliable transport protocol.

- Missing features: security, low-level semantics for MIBs, protocol primitives, information model and method call.

By use of Web technologies, WIMA solved most problems among the listed problems. WIMA proved that the push-based network management is more appropriate than pull-based network management. A WIMA-based research prototype, JAva MAnagement Platform (JAMAP), implement push-based network management using Java technology. WIMA gave a way for exchanging management information between a manager and agent using HTTP. HTTP messages are structured with MIME multipart. Each MIME part can be an XML document, binary file with BER-encoded data, etc. By dissociating the communication and information model, WIMA allows management applications to transfer SNMP, CIM or other management data. The dissertation made another core contribution in showing that XML is especially convenient for distributed and integrated management, for dealing with multiple information models and for supporting high-level semantics. We have also employed push-based network management on our architecture and SNMP-to-XML model-level mapping translator for integrating SNMP agent into our architecture.

3. Web-based Element Management Architecture

The purpose of this chapter is to explain our research on Web-based element management using embedded Web server (EWS). We present our research to develop an efficient and lightweight EWS for Web-based element management. We propose an architecture of embedded Web server that provides a simple but powerful application interface for network element management. A developer can choose an appropriate interface mechanism depending on the management information characteristics or types of Web documents. An integration mechanism between a Web document and a management application enables developers to add new management functionalities by merging Web documents with management application programs that generate specific dynamic management information.

3.1 Embedded Web Server Architecture

We have designed an EWS that consists of five parts: an HTTP engine, an application interface module, a virtual file system, a configuration module, and a security module. The design architecture of our EWS is illustrated in Figure 3.

The most important part of the EWS is an HTTP engine, which serves a client's request. The minimum requirement for an HTTP engine is that it must be compliant with HTTP specifications [10,16] for communicating commercial Web browsers. Unlike general Web servers that start a new thread or process whenever a new connection is made, normally an HTTP engine supports multiple simultaneous users while running as a single process. The number of processes that the server requires can impact on both RAM usage, due to the stack space per task, and CPU usage. In the next subsection, we explain an HTTP transaction

process using a state transition diagram.

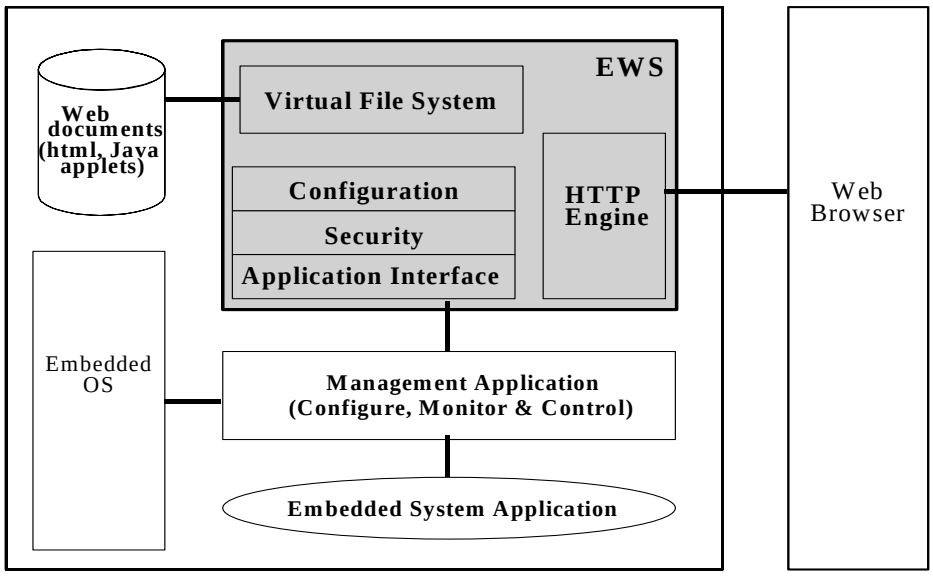


Figure 3. EWS-based Element Management Architecture

In an EWS, the application interface module enables developers to add new management functionality. With any off-the-shelf Web authoring tool, it can merge Web documents with management application programs to generate specific dynamic management information. CGI and SSI type interface, which is introduced in Section 3.1.2, are provided for interacting with the embedded application. CGI type interface is used for generating Web document based on parameters submitted by operator through Web browser. And SSI type interface is appropriate in dynamic Web page generation. Details about application interface are described in Section 3.4.

The virtual file system (VFS) provides the EWS with virtual file services, which are file_open for opening the file, file_read for reading the file, and file_close for closing the file after reading. The file system has a data structure storing file information such as file size, last modified date, etc. The data structure for an HTML documents file needing dynamic information must store the pointer

of the script and the function name called by the script. To construct this VFS we need a Web compiler. The Web compiler supports any format, such as Java, GIF, JPEG, PDF, TIFF, HTML, text, etc. It compiles these files into intermediate C-codes and then compiles & links them with the Web Server codes. The resulting structure does not require a file system, yet the files are organized like in a file system - a virtual file system. The Web browser traverses this virtual file system just as if it were an actual file system. Figure 4 illustrates the process of a Web server making a virtual file system. Further details about the preprocessor tool, Web compiler, are provided in Section 3.5

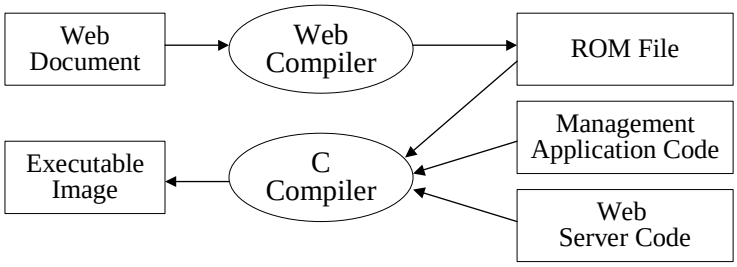


Figure 4. Process of a Web server making a Virtual File System

Security is an important concern in network management. Therefore, an EWS generally has a security and/or configuration module. Security is accomplished by defining security realms on a server and username/password access to each realm. When a request comes in for an object in a protected realm, the server responds with a response code of 401 (Unauthorized). This will force a browser to prompt the user for a username/password pair. The original object request will be resubmitted with the username/password, base-64 encoded, in the request header. If the server finds the login correct, then it will return the requested object, otherwise, a 403 forbidden response is returned.

The configuration module provides the administrator with the functionality to set the embedded Web server configuration from any standard Web browser. The configuration environment variables passed at startup define

the number of concurrent connections, socket port, own host name, root file path, default “index”, inactivity timeout and time zone. Common usage of Web browsers makes it a more important matter to protect abnormal access to the sensitive information of network devices, especially those that involve equipment configuration or administration.

3.2 EWS Process Structure

We designed an EWS as a finite state machine (FSM), which processes an HTTP request in a sequence of discrete steps. Figure 5 shows the state transition diagram of the HTTP engine. In order to support multiple connections in a single thread environment, multiple finite state machines are run by a scheduling system which uses a lightweight task structure. It consists of a pointer to the function being run, a variable holding the state in the FSM, and a flag indicating whether the FSM can be run or blocked. The scheduling system allocates an available FSM for an accepted connection, checks each FSM to see if it is blocked or runable and, if it is runable, moves the FSM one step.

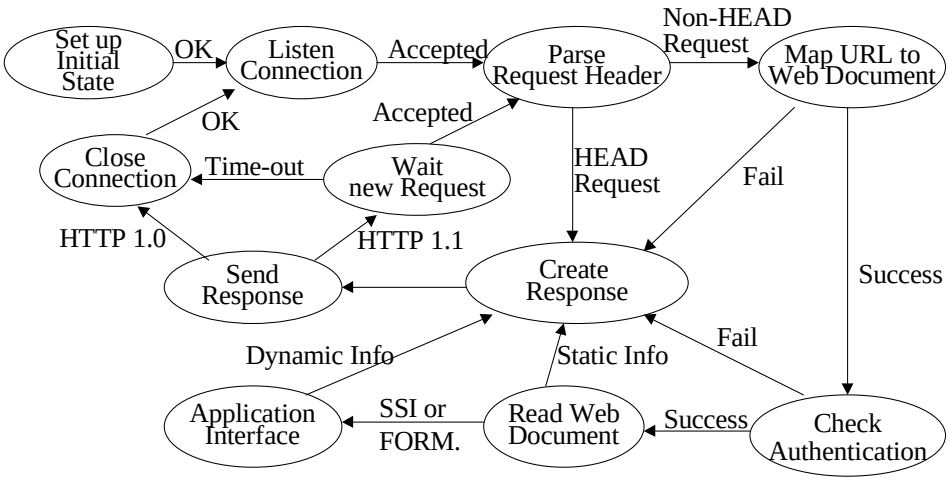


Figure 5. EWS Finite State Machine

Each state in an FSM can check for the presence of data that is ready to be processed at the entry point; if none is ready, the FSM can block itself until data arrives. When data becomes available at the entry point, the FSM can then be unblocked so its handler can perform the task of state, and turn over the result to the next state by changing the state flag and pointer to the handler.

The following list describes the behavior of each state.

- *Set up Initial State*: Set up the task structure for an FSM. The task of this state is performed at the server initial time for all FSMs.
- *Listen Connection*: Check to see if any request is allocated to this FSM.
- *Parse Request Header*: Read the HTTP message, parse the HTTP header and store the parsing result.
- *Map URL to Web Document*: Determine type of application interface and store a pointer to the handler.
- *Check Authentication*: Force authentication of the user upon the URL and user name/ password.
- *Read Web Document*: Read Web document from virtual file system.
- *Application Interface*: Call application function upon the URL.
- *Create Response*: Create HTTP response header.
- *Send Response*: Send HTTP header and Web document.
- *Wait new Request*: Wait for a new HTTP request from the same TCP connection if the received request says HTTP/1.1 support.
- *Close connection*: Close the TCP connection.

The approach of EWS as a finite-state machine increases portability because the EWS can be ported into a system that does not support multi-thread or multi-process task. Another advantage of the approach is that computing resource used by EWS is saved in terms of memory and CPU. It is unfair to say that finite-state machine approach is more efficient in computing resource usage than multi-thread approach in supproting concurrent input/output in a single process. Because the computing resource usage of finite-state machine is totally

dependent on developer's design of the machine. But well-designed finite-state machines uses less computing resource than multiple threads that consists of a program counter, register set and a stack space. The designed finite-state machine of EWS is such a small that it can be implemented into a sufficiently efficient single thread in computing resource usage. In the section 5.1, we validate the efficiency of the approach by analyzing implementation result of the designed state machine.

3.3 Extended Embedded Web Server Architecture

As mentioned in Section 3.1.2, only the scheme of HTTP and HTML is client-driven. So this scheme is not suitable for displaying real-time data and reporting event asynchronously. Java applets can display dynamically management information within the Web browser if it can communicate directly with embedded Web server.

Simple Network Management Protocol (SNMP) [6] is the most widely used management framework for managing network devices on the Internet. Its protocol is simple enough that it can be implemented in small platforms without much difficulty. Now most network devices are equipped with an SNMP agent. With integration of SNMP and the Web-based element management architecture, the advantages of Web-based element management are preserved without the giving up the SNMP implementations.

Figure 6 illustrates the extended architecture for Web-based element management architecture. Java implementation of SNMP mediates between an SNMP agent and a Web browser. The Java SNMP source code is written and compiled to produce a Java SNMP applet. This applet is stored in a network device and is transferred by the EWS to the browser over the network at run time. After loading on the JVM of a browser, the Java SNMP applet communicates with the SNMP agent in the network device and enables the administrator to control

and monitor the network device through the browser, using SNMP messages. In addition to the Java SNMP applet, the network device in this scenario must store at least one HTML document containing reference to the applet. The HTML document is loaded into the Web browser and then the Web browser would automatically request the Java SNMP applet referenced by the previous HTML.

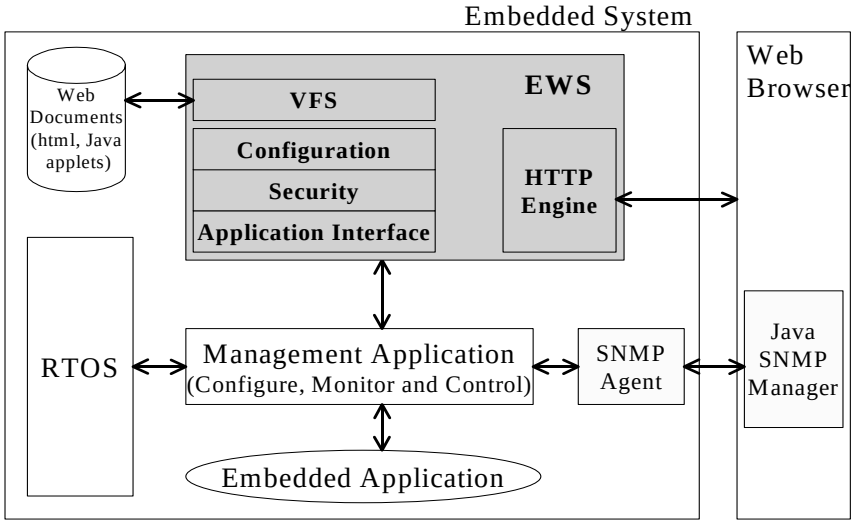


Figure 6 Extended EWS-based Element Architecture

The code size of a Java SNMP applet is small enough to be embedded into the network device because SNMP is simple (three basic massage types and a simple message format) and light protocol (uses UDP as its transport protocol, and thus does not have connection setup and acknowledgement overhead). SNMP defines an alert message and traps, which can be directed toward one or more trap receiver stations. If a trap management application is implemented as the Java SNMP applet and loaded from the network device, traps can be collected and viewed together by the Java SNMP applet where appropriate responses will follow.

3.4 Management Application Interface Mechanism

Management information can be classified on the basis of update period, direction of information flow or object of information source. From the viewpoint of update period, some management information changes dynamically, and some does not change at all. Further, some information possesses real-time characteristics, for example a dynamic graphic form in a real-time update or event notification to perform some action. Regarding the direction of information flow, some information can originate from a Web browser and travel to a Web server and vice versa.

As depicted in Figure 6, a management application includes a Web Management API that is used by EWS exclusively, and SNMP agent that has standard SNMP interface. These two elements can be management information source for providing Web-based element management user interface. Also, different types of Web documents have very distinct characteristics, especially between HTML and Java. All things considered, it is more desirable to provide several effective interface mechanisms between the Web document and management applications. This variety in determining the interface mechanism enables developers to choose an appropriate mechanism on the basis of characteristics of management information and Web documents. We define four interface mechanisms, which are depicted in Figure 7, (**a)**: CGI-Type, (**b)**: SSI-Type, (**c)**: SSI SNMP-Type and (**d)**: Java SNMP-Type. Each mechanism has its own advantages over the others. They are explained and compared in the following subsections.

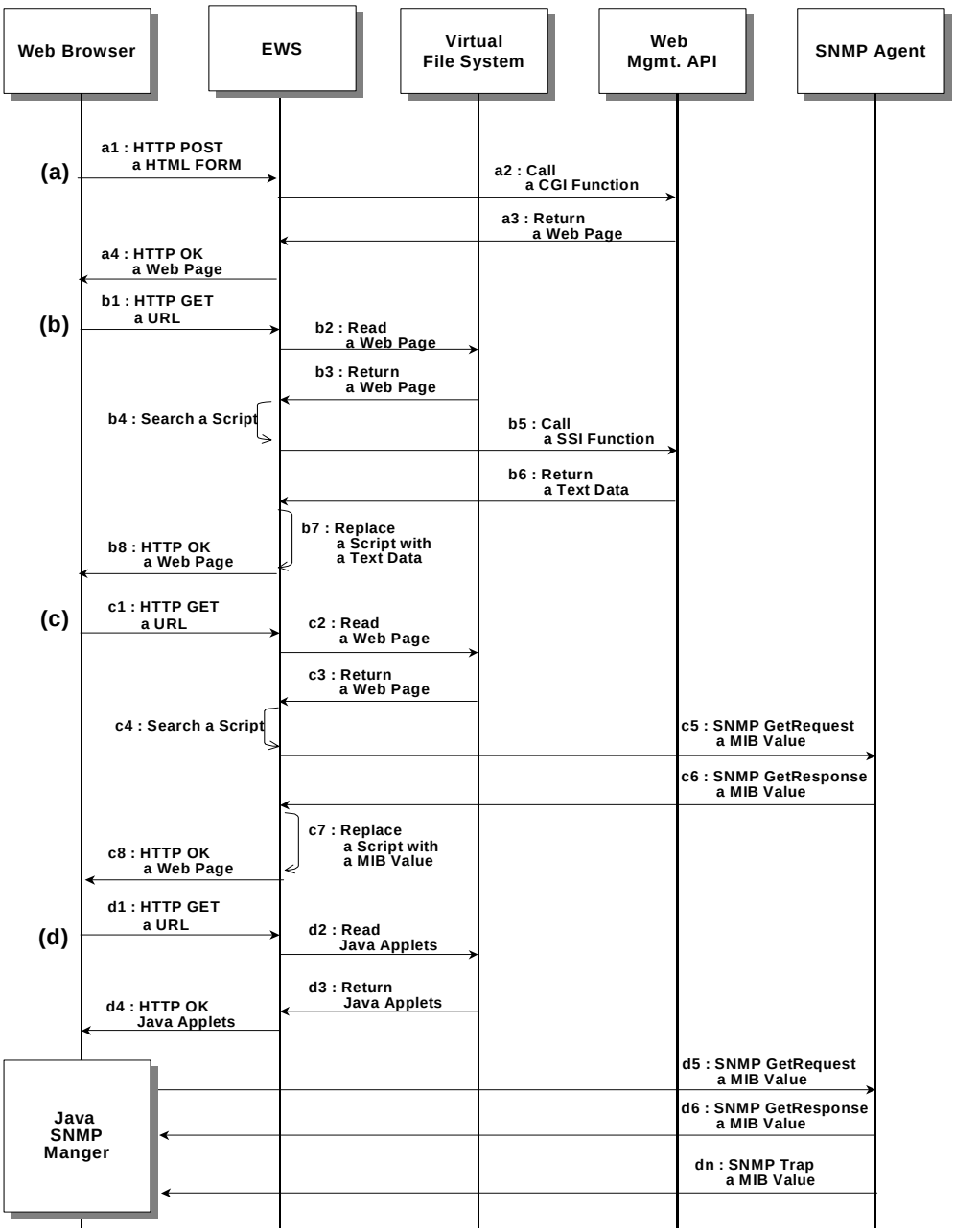


Figure 7. Management Application Interface Mechanism

3.4.1 CGI-Type Interface

CGI provide a mechanism for the management application to generate and serve Web pages on the basis of HTML form data submitted by the browser. Each URL is mapped to function call that generates the Web page. The (**a)** arrow in Figure 7 shows the data flow of the CGI-Type interface mechanism.

While this approach is perhaps the easiest for embedded Web server developers, it is by far the most difficult for embedded application developers because CGI scripts are tedious to write. Once written, the display of the Web page can only be determined when the script is executed. In an embedded system, this implies building the executable code, downloading it into flash memory, and booting the device before the Web page can be viewed by a browser. Therefore, CGI solutions require a long development time and are difficult to maintain.

But for management applications that process the user commands that have much parametric information and produce simple Web pages to contain only success or fail information, this mechanism is a good solution because the interface mechanism is simple. No special integration mechanism between a Web document and the management application is required - just a few useful library functions are enough to integrate them.

3.4.2 SSI-Type Interface

As introduced in Section 3.1.2, SSI is another mechanism for generating Web pages dynamically. Embedded Web server interprets Web documents before sending it to search marked-up tags within the Web page. The server maintains a database for mapping script names of marked-up tags into management application functions. The server replaces the tags with return value of management application function. In a general Web server, parameters can be passed to the Server Side Include routine by the one of two methods: in the URL or in the Web page. We exclude the parameter passing of SSI in the URL because of security problems [50].

The (**b)** arrow in Figure 7 shows the information flow of this interface mechanism. An embedded system initialization code is used to register the server-side scripts with the function calls of the management application. The code for a function call is invoked when the server interprets server-side scripts. Since SSI is a substitution mechanism, where text is inserted into a document at tagged locations and CGI is a generation mechanism, where the whole Web page is produced by a real program, CGI shows higher flexibility than SSI. But SSI requires little development expertise beyond writing the Web page and also has a lower development time than CGI.

However, interpreting scripts at runtime in an embedded system may impact system performance. Moreover, a significant amount of memory is required to maintain a database for mapping script names into embedded software functions and variables. In order to offload such substantial Web server processing from the embedded system at run time, a special integration mechanism, a preprocessing, is used. Further details about the preprocessor tool, Web compiler, are provided in Section 4.5.3. Where a Web document displays status information in the form of a table or other formatted style, this interface mechanism is an ideal solution because complicated display formats can be separated from the management information production method, which is almost system dependent. Conversely, information production logic is independent of its usage for display. A typical and proper use of this interface mechanism is with state report generation. The marked-up tags are contained within standard HTML comment elements. These elements are blocks of text surrounded by the **<!--#** and --> sequences.

3.4.3 SSI SNMP-Type Interface

The SSI-Type interface mechanism does not restrict the form of management applications; it is just a set of function calls. When the management application is an SNMP agent, a more elegant and automated method can be introduced. The SSI SNMP-Type interface mechanism is a specialized case in limiting the form of management application to an SNMP agent. Basically, the

development process and integration mechanism are the same as those of SSI-Type mechanism, but the management application functions called by the script interpreter are specialized to the interface provided by an SNMP agent.

There are two ways of interfacing with the SNMP agent; one of them is to use the local loop interface of an SNMP stack and the other is to use the SNMP MIB implementation directly, which is more efficient in resource usage. The direct interface to the SNMP MIB implementation depends on the implementation of the SNMP agent. But the local loop interface of the SNMP stack provides a standard interface just like a remote access. Therefore, this method has higher portability and reusability than the direct interface to the SNMP MIB implementation. The SNMP protocol itself is simple and light so the performance overhead of the local looped SNMP stack method can be negligible, especially in case of SNMPv1. The local looped SNMP stack method is used for interfacing with an SNMP agent using the SSI-Type mechanism.

Because the SNMP agent is used as a management application, the management information is also defined in SNMP SMI. This well-defined management information makes it easy for a developer to understand management information to be appeared in Web pages, while formalized specification method is not supported in CGI-Type or SSI-Type interface mechanism. All network systems deployed at this time are equipped with an SNMP agent, so the use of a legacy SNMP agent can reduce the development time of a Web-based user interface.

A Web-based SNMP MIB browser is the most common use of SSI SNMP-Type interface mechanism. The browser gives the traversal functionality on the defined MIB tree for investigating the up-to-date values of SNMP MIB variables. The Web interface for the MIB browser takes the typical form of a user interface. Accordingly, the Web pages of the browser can be produced automatically from SNMP MIB definition.

The MIB2HTML compiler [35] reads SNMP MIB definitions and generates Web pages of the MIB browser. Marked-up tags in order to retrieve or

set MIB variables at run time are inserted into the generated Web documents. A detailed explanation about MIB2HTML is provided in Section 4.5.3. The management application interface is identical for all MIB definition by use of local loop interface of a SNMP stack. Therefore the interface program can be supported in a form of library code. This means that Web-based MIB browser of the network device embedding an SNMP agent can be developed without an additional management application program or any other sort of program. Only Web pages generated by MIB2HTML is added to the embedded system.

3.4.4 Java SNMP-Type Interface

The Java SNMP-Type interface supports extended architecture of Web-based element management architecture. An SNMP stack is included in the Java applet as well as the manager function. The code size of a Java SNMP applet is small enough that it can be embedded into the network device (30 Kbytes in compressed format) because SNMPv1 is simple (three basic message types and simple message format) and light (uses UDP as its transport protocol, and thus does not have connection setup or acknowledgement overhead). SNMP defines traps that can be directed toward one or more trap receiver station. If a trap management application is implemented as a Java SNMP applet and loaded from the network device, traps can be collected and viewed together with a Java SNMP applet, and appropriate responses taken.

3.4.5 Comparison of Application Interface Mechanisms

In this subsection, the characteristics of previously described application interface mechanisms are compared. We assume that the SSI SNMP interface mechanism implements the SNMP MIB browser. Table 1 summarizes a comparison of the interface mechanisms. Unlike programming support on a general-purpose computer, programming support on an embedded system is so limited that in most cases programmers cannot use advanced shell or script programs such as csh, Perl and Tcl/Tk. Building blocks of system program are

written in C or C++.

	CGI-Type	SSI-Type	SSI SNMP-Type	Java SNMP-Type
Web documents development method	Management Application Program	Web Authoring Tool + Marked-up tags insertion	MIB2HTML Compiler	Java applet program
Management application programming	Necessary	Necessary	Unnecessary (Library code)	Unnecessary (SNMP Agent)
Management information source	Embedded Management Application	Embedded Management Application	SNMP Agent	SNMP Agent
Event Support	No	No	No	Yes
Web documents development cost	High	Low	Very Low	High
Network load / Web page	1 HTTP requests	1 HTTP requests	n-SNMP & 1-HTTP	1-HTTP & Continuous SNMP
CPU Load	Small	Medium	Large	Medium
Code size	Management Application Program	HTML + Management Application Program	HTML + Management Application Program	Java class
Portability	Low	Middle	Middle	High

Table 2. Comparison of Interface Mechanisms

Each interface mechanism has a different development method for Web documents. For the CGI-Type mechanism, the management application program for the Web interface generates Web documents; consequently, its development cost is high. For the SSI-Type, Web documents are developed using a Web authoring tool and marked-up tags are inserted into the Web pages. Compared with the CGI-type, its development cost is low because Web authoring tools are used. For SSI SNMP-Type in the case of SNMP MIB browser implementation, MIB2HTML compiler generates whole HTML pages for the MIB browser, so its

development cost is negligible. For Java SNMP-Type, Web documents are developed in Java applet classes on the supported Java SNMP stack. Its development cost is high.

In the case of wanting to add new management functionality using the CGI-Type and SSI-Type, it is necessary for a developer to add a new Web interface module of management application program that generates or supports HTML, respectively. However, when using SSI SNMP-Type and Java SNMP-Type it is unnecessary to add a new management application on the assumption that a legacy SNMP agent supplies the necessary information.

With resource usage, the CGI-Type uses the least amount of CPU, memory and network resources because an EWS only calls on a CGI function just once and sends the whole result text of the function call without any processing. Compared with CGI-Type, SSI-Type uses more CPU resources than the CGI-Type because it needs to search the script of marked-up tags and function calls and text replacement on each marked-up tag. Further, SSI-Type uses a file system for storing HTML documents while the CGI-Type does not use any type of file system. SSI SNMP-Type uses the largest amount of CPU, memory and network load because it needs to use an additional local looped SNMP stack for the SSI style. A Java SNMP-Type mechanism does not require script parsing, but the Java SNMP manager that is downloaded from the embedded system to a Web browser issues SNMP requests continuously. Compared with CGI-Type, Java SNMP manager downloading uses less computing resources than generating Web page by CGI-Type, but continuous SNMP communication uses computing resources after downloading.

Management application of CGI-Type interface is programmed by system programming language, not well-known script language, and it uses EWS interface library functions that are supported by the embedded Web server, not a well-known interface mechanism such as standard input/output in UNIX. Its portability is marginally low because the CGI-Type management program depends on system environment and library functions of the embedded Web

server. In the SSI-Type interface, at least portability of Web pages is guaranteed, but SSI script programs to be executed by EWS have the same portability as CGI-Type interface programs. The SSI SNMP-Type management application has the same level of portability with the SSI-Type management application because two interface mechanisms are the same. The Java SNMP manger is just stored at the embedded system, runs on the Web browser and communicates via SNMP protocol. Its portability is high because there is no dependency on the Web server or on the embedded system. Since only Java SNMP-Type supports asynchronous communication, it is possible to implement event notification with only Java SNMP-Type interface mechanisms.

3.5 Management Application Integration Mechanism

An integration mechanism between Web documents and management applications enables developers to add new management functionality by merging Web documents, which provides a user interface format with management application programs that generate specific management information. Through the integration mechanism, Web document development can be separated from management application development. Consequently, the management application developer no longer needs to consider the user interface of Web documents through program logic.

Also, Web document prototyping, involving different user interfaces with a deployed Web document, no longer requires code changes and recompilation of the management application program if the same management application interfaces are used and if a Web page can be uploaded dynamically. For rapid development, an easy but powerful integration mechanism must be provided.

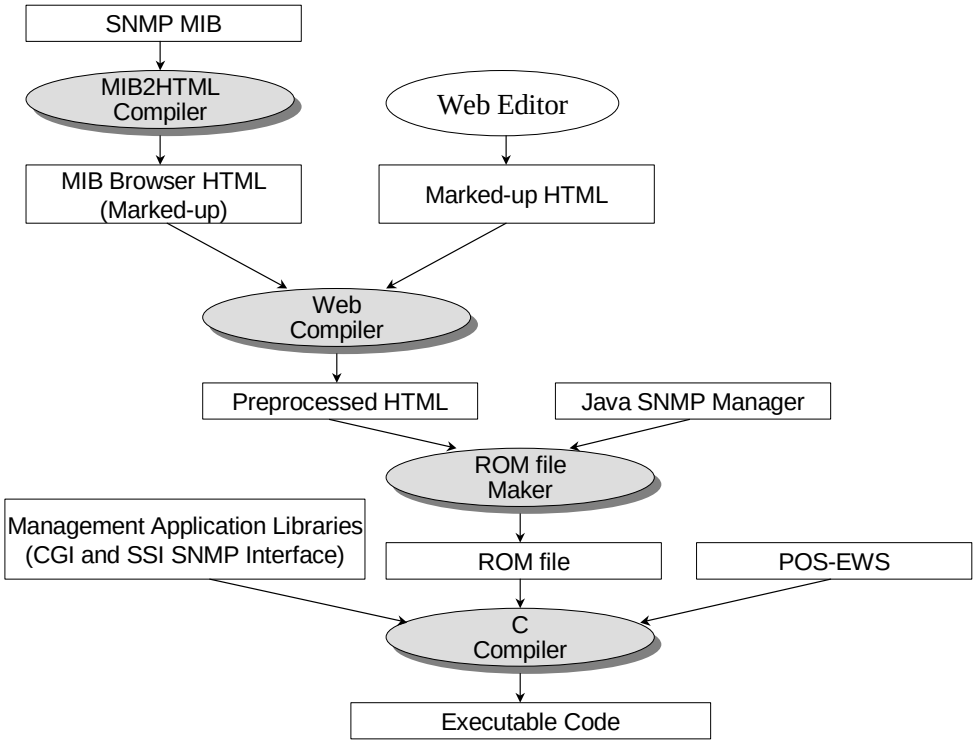


Figure 8. Management Application Integration Mechanism

The integration mechanisms are depicted in Figure 8. There are two compilers, MIB2HTML [35] and a Web Compiler [36], and a C cross-compiler for the embedded system. The compilers are used for accelerating Web document development time. The MIB2HTML compiler generates marked-up HTML files from the SNMP MIB definition for the SNMP MIB browser and the Web compiler processes the marked-up HTML file to be used by embedded Web server at run-time. The ROM file maker generates binary coded ROM files for supporting the virtual file system, where embedded OS does not support a file system. Binary coded ROM file consists of character arrays and the virtual file system is a limited set of read-only functions built into the ROM. The character array is the result of simple conversion and compression from plan source file to

C language array.

The management application includes user-programmed function for CGI-Type and SSI-Type interface and library functions supported by embedded Web server for SSI SNMP-Type interface. This management application program is combined with embedded Web server source code in order to generate an executable code. The code is executed in a single thread process in the embedded system.

3.5.1 CGI Library

In the CGI-Type application interface mechanism, the management application program generates a Web document directly. Therefore, the presentation logic on the Web browser is combined with management information production logic in the management application program. The best way of integration is to provide the management application developer with an easy method for Web document generation. As mentioned before, a typical application of the CGI interface mechanism is user command processing such as setting system time, and simple display of the processing result through Web interface. Frequently used functions are to parse the parameters and to generate simple Web documents.

Frequently used functions in the CGI interface can be offered in the form of a library. The defined library functions are listed below:

- *CgiGetVar*: Look variable up in the table of HTTP tags. Return the value of a variable.
- *CgiGetCookie*: Get value of cookie from the HTTP tags.
- *CgiError*: Return an error message to the requesting browser.
- *CgiHeader*: Output the common HTTP header and HTML tags to begin request response.
- *CgiWrite*: Write formatted data back to the user’s browser.

- *CgiSetCookie*: Issue cookie to the browser.
- *CgiFooter*: Outputs the standard HTML tags to finish a web page.
- *CgiDone*: Close the connection to the browser when completing a request.

3.5.2 Web Compiler

In the SSI-Type interface mechanism, while Web documents can be quickly prototyped with readily available desktop authoring tools, the prototype must then be integrated into the system software. Embedded Web server works with a fixed set of integrated Web documents that is usually frozen at the time the embedded system is manufactured. Next, it incorporates dynamic information of management application at run-time in such a way of interpretation and execution of the script. Since Embedded Web server has no information about the location of script in the HTML file, it scans the whole HTML file to find script tags. It also has to look up the script function table in order to search for the function pointer to be executed.

The Web compiler can offload embedded Web server's scanning of whole HTML file and script table lookup. The Web compiler preprocesses the marked-up HTML file and records the position of tags and its function pointer with the HTML file. At run time POS-EWS reads and sends the HTML file before the starting point of a tag, continuously calls the script function, and sends the return result of the called functions, and then proceeds repeatedly with the same procedure for the remaining HTML. POS-EWS does not search for script tags or script functions in HTML file and script function table, respectively. The Web compiler extracts script information from Web pages and generates C code. The C code is then compiled and linked with POS-EWS and the management application program to produce a tightly integrated executable code. The Web compiler enables sophisticated dynamic Web-page capabilities by performing complex tasks up front and generating an efficient and tightly integrated representation of

the Web pages and interfaces in the embedded system.

3.5.3 MIB2HTML Compiler

When the SSI SNMP-Type interface mechanism is used for implementing SNMP MIB browser, the MIB2HTML compiler converts MIB definitions into a set of HTML files that are used for SNMP MIB browser. The HTML pages automatically include the marked-up tags used by embedded Web server to display and traverse SNMP information on the basis of the MIB tree structure. HTML files are created for every MIB table. The root file generated contains hyperlinks to all the HTML files for the MIB table. This compiler interprets MIB tables, and automatically detects and fills in enumerated types. It extracts the MIB descriptions and inserts them as online help automatically.

When embedded Web server receives an HTTP request from a browser to display the HTML pages of the MIB browser, it loads the HTML page into memory from the corresponding ROM file. This file is used by embedded Web server to produce the HTML containing up-to-date value of SNMP MIB variable. It does this through the embedded Web server script handler. The handler calls the appropriate function for each markup tag, which is inserted by the MIB2HTML compiler. Once all the marked-up tags have been replaced, the resulting HTML is returned to the browser. This interface mechanism is the same as the SSI-Type, but functions called by the script handler have been made for exchanging management information with the SNMP agent by the use of a local SNMP stack.

The general format for a script is

<!--#***snmp command args*** -->

where ***command*** is a command and ***args*** are optional space separated arguments to the command. Here are some examples of scripts:

- *<!--#snmp community-->* : Display the current community name used for interfacing with the SNMP agent.
- *<!--#snmp set oid type -->* : Return an HTML FORM structure

including button and input text box for setting SNMP MIB variable.

- *<!--#snmp get oid -->* : Display the up-to-date value of the SNMP MIB variable.

3.5.4 Java SNMP Manager Library

In the Java SNMP-Type interface mechanism, the Java applet is a Web document and SNMP agent is a management application. The Java SNMP interface enables developers to add new functions on top of the SNMP stack in the Java program. The Java SNMP manager sends a SNMP request to the managed element. Strictly speaking, the added function is a subset of the manager role from the viewpoint of network management. The management function of the Java applet manager has limitations in performing complex tasks because the code size of the Java applet must be small so that the code could be downloaded in a reasonable time. Another limitation of the Java applet program is that Java applets are restricted in accessing local disks or executing another program, moreover communicating with the Web server except HTTP by the Java applet security model (sandbox) [52]. We used code-signing mechanism for the Java SNMP manager applets to remove those restrictions. Real-time data displays without logging and alarm notifications without history are typical tasks of Java applet manager.

Similar to the CGI interface mechanism, the best method of integration is to provide developers with frequently used library functions when programming the manager in Java. As mentioned before, a typical and suitable application of Java SNMP interface is a real-time data display in a graphic chart or an alarm display with a simple alert. An example of implementation is as follows. Java SNMP manger sends a request for interface in/out octets of interfaces group to SNMP agent, and traffic flow graph can be drawn using a library. Like the Java SNMP stack, the code size of this library is so small that the upload time from the Web server is negligible: a total of 30 Kbytes.

4. Web-based Network Management Architecture

We extend the Web-based element management architecture to the XML-based network management architecture. The extended architecture uses HTTP and XML as communication protocol and management information encoding, respectively. In this section, we describe how they can be used in defining management protocol and management information. Further, a management system on the basis of our proposed architecture will be developed to support basic network management functions. We will define a message format and exchange procedure for transferring management information between Web-based management applications.

4.1 Web-based Network Management Models

Generally a management architecture consists of three models; information model, communication model and organization model. In this section, we describe Web-based network management architecture with respect to the three models.

4.1.1 Management Information Model

The management information model defines the modeling approach such as entity relationship, data type approach or object orientation. It also defines a unique notation for describing the management information. We decided to use XML schema for management information model. The reasons why we do not rely on standard information model such as SNMP SMI or CIM schema are as follows.

First, we must define new management information. As mentioned in

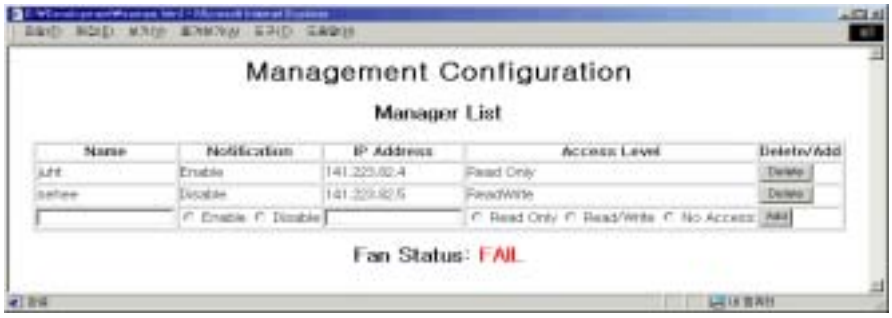
Section 1, our target is to enhance the Web-based element management to be understandable by WBM manager. Most management information provided by Web-based element interface is not yet defined in the standard information model. This means that we must define new management information. Therefore, it is not unusual to choose appropriate one among the information modeling methods.

Second, information modeling using XML schema is a wide-spread approach in other application areas. XML lets you model content that ranges from book-oriented document types to commercial transactions. XML is an enabling technology for business-to-business integration, data interchange, e-commerce, and the creation of application-specific vocabularies. Database developers are particularly interested in XML as a building block for creating middle-tier servers that integrates data from disparate databases.

Third, compared with SNMP SMI and WBEM CIM, XML schema has many advantages in addition to advantage of using XML in network management. XML schema is easy to learn. Developers can use many powerful and convenient XML editors for creating XML documents. Another advantage is that XML data is concise and easy to read because of no need for translation. It is prerequisite to translate from SMI, CIM or UML to XML if modeling and encoding techniques are different, for example, SMI vs. XML and CIM vs. XML. Generally translation results in verbose XML data and low readability. Another advantage is that XML can be utilized for various purposes including validation. By use of XML authoring tools, developer can generate sample XML data, database schema and other forms of data structure based on XML schema.

Figure 9 shows an example of simple management information modeling from Web browser display provided by Web-based element management. The network device has a table of manager that has attributes of its name, receipt of notification, IP address and access level. Operator can add or delete an entry to/from the table. The fan status also can be monitored through Web-based element management. In the example, the title of data and operation correspond XML element and XML attributes respectively and the structure of information is

still preserved in XML schema. By adopting some simple conventions, XML schema can successfully model management information displayed on Web browser through Web-based element management. By capturing data type, key relationships and structure, developers can produce easily XML schema by use of an XML editing tool.



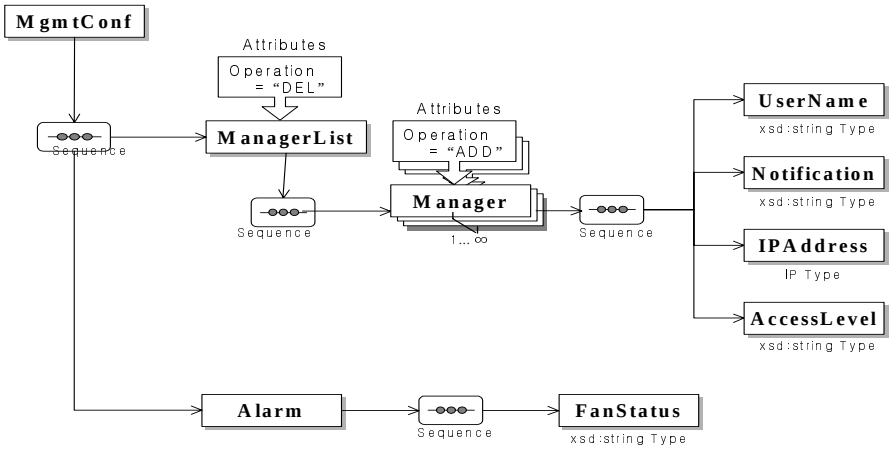


Figure 9. Information Model Example

4.1.2 Management Communication Model

Management entails configuration, monitoring and control of potentially

physically dispersed resources. An intrinsic part of this process is the exchange of management information. The communication model defines the concepts for this exchange of information between management applications. We limit management application to WBM manager and WBM agent in our physical architecture mentioned in Section 1.

Communication model must deal with the specification of services and protocols for the exchange of management information. Also, it must define the syntax and semantics for the communicative data structure. With respect to service and protocols, Web-based network management architecture follows the original model of structuring data over HTTP without any extension. And it uses XML for the management information encoding syntax. This means management data is transferred over HTTP payload in a form of XML document.

For notification delivery, the communication model must provide asynchronous communication method. However, HTTP is strictly request-response protocol from a client to a server. This means WBM agent cannot send an event message to WBM manager asynchronously. One of the WIMA solutions to the problem is to add HTTP client to an agent and HTTP server to a manager. Web-based network management architecture utilizes the solution of WIMA.

Another important issue to communication model is addressing of managed objects. When a manager requests management information, it must specify a unique name of managed object to be retrieved. Web-based network management architecture takes XPath standard for addressing managed objects. This solution offers following advantages. First, it is not unusual because XPath is a standard for addressing parts of an XML document. XPath is already used in conjunction with well-known XSLT style sheets [30,31] to identify pieces of an XML document targeted for transformations. Accompanying advantages are large available implementations and broad support from application programs.

Second, WBM manager can query effectively on managed objects of WBM agent. XPath expressions are formed using element names, attributes and built-in functions. Given the hierarchical nature of XML, one of the simplest

expressions is to follow the tree structure to point to a particular element. For example, if a WBM manager makes a request with the expression "/MgtConf/ManageList", the WBM agent would extract all the <Manager> elements and send them to WBM manager in the section 3.1 example. Using the // operator, you can retrieve an element regardless of its position in the hierarchy. The character "*" is used as the wildcard. Also the format: //element_name[@attribute_name] is used for selecting an element based on its attribute. By the use of "//*[@Operation=ADD], WBM manager can look for all elements that have an attribute named "Operation" with value of “ADD” regardless of its position. There are a number of functions a WBM manager can use with XPath expressions. A rich addressing mechanism give a manager the ability for making efficient query.

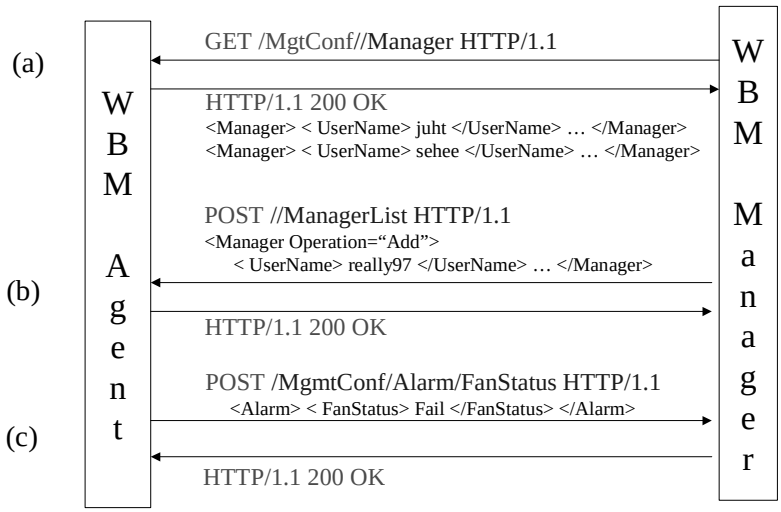


Figure 10. Web-based Network Management Communication Example

Figure 10 is a data exchange example between a WBM agent and manager. In this example, Figure 10 (a) illustrates a monitoring interaction that a WBM manager requests all registered “Managers” to a WBM agent. Figure 10 (b)

illustrates a control interaction that a WBM manager adds a “Manager” to the “ManagerList” in a WBM agent. Figure 10 (c) illustrates a notification interactions that a WBM agent sends asynchronously to a WBM manager an alarm that “FanStatus” changed into “Fail”.

4.1.3 Management Organization Model

The organization model of a management architecture defines the actors, their roles and the fundamental principles of their cooperation. The most well-known organization model in the network management area is the manager-agent paradigm. There have been many proposals for new organization model such as management by delegation, policy-driven management, push-based network management, mobile and intelligent agent. Web-based network management architecture does not indeed include new organization model, but we integrated push-based network management into Web-based network management architecture on the conclusion that the model is essential for efficient network management.

Push technologies are concerned with automated information delivery based on the publish/subscription/distribution paradigm, sometimes using true push (server-initiated communications) but more often than not using some form of a scheduled pull. There is a multitude of Web push applications: stock quotes, live game score updates, etc.

There are many methods for Web push from HTML refresh meta-tag [9] to Java applet [11] and CDF [53]. Applying these current incarnations of Web push to Web-based network management architecture has following problems. First, they are inefficient in use of bandwidth because they are not true push. Second, they rely on display technique such as HTML and Java applet.

We incarnate push-based network management into XML as following. Management information model by WBM agent has already been published in Web-based network management architecture because XML has a self-describing

capability. And WBM manager can do discovery management information model by just retrieving parts of management information. With respect to communication model and information model, distribution of management information and notification delivery are the same: asynchronous communication and no additional information model. But subscription is different with control, monitoring and notification of manager-agent paradigm. With the information model viewpoint, subscription information must be added to the original information specification. We define a new information specification for subscription in a form of XML schema. WBM manager sends subscription information conforming the new information specification. After receiving the subscription information, the WBM agent schedules a series of message distribution and sends it to the subscriber according to the schedule.

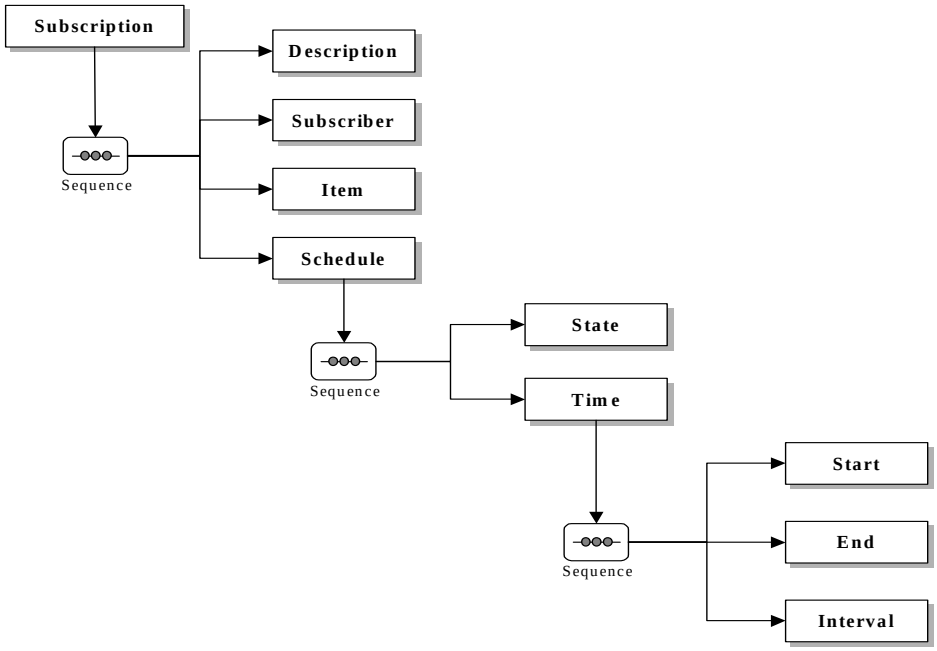


Figure 11. Subscription Information for Push-based Management

XML schema for subscription is depicted in Figure 11. Subscription information includes description of itself (Description), subscriber's URL for receipt (Subscriber), managed object's XPath expression (Item) and trigger information (Schedule).

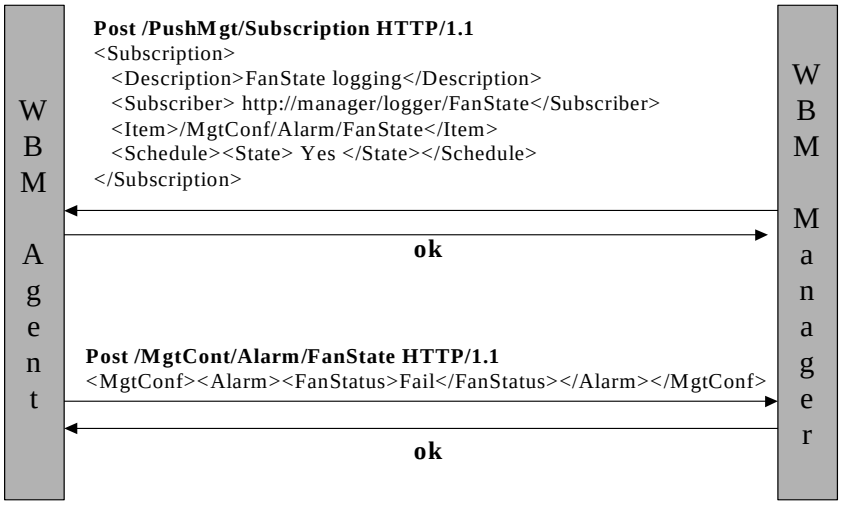


Figure 12. Communication Examples for Push-based Management

Communication example for subscription and distribution is depicted in Figure 12. In the first message, a WBM manager requests a message to the URL, http://manger/logger/FanState, when the state of the managed object having address /MgtConf/Alarm/FanState changes. In the second message, a WBM agent notify the state change of subscribed managed object.

4.2 Web-based Network Management Platform

In this section, we describe the network management platform which supports XML-based Network Management (Web-based network management architecture) architecture. As mentioned in section 1, Web-based network management architecture has two key components: WBM agent and WBM manager acting as an agent and manager, respectively. In section 4, we provide a Web-based element management architecture having an embedded Web server as a core component. We extended the embedded Web server to Web-based network management architecture platform by adding XML functionalities.

4.2.1 WBM Agent

Figure 13 shows structure of WBM agent. DOM tree, XPath Handler, Push Scheduler and HTTP Client Engine are added components to element management architecture, and they are shaded in the figure.

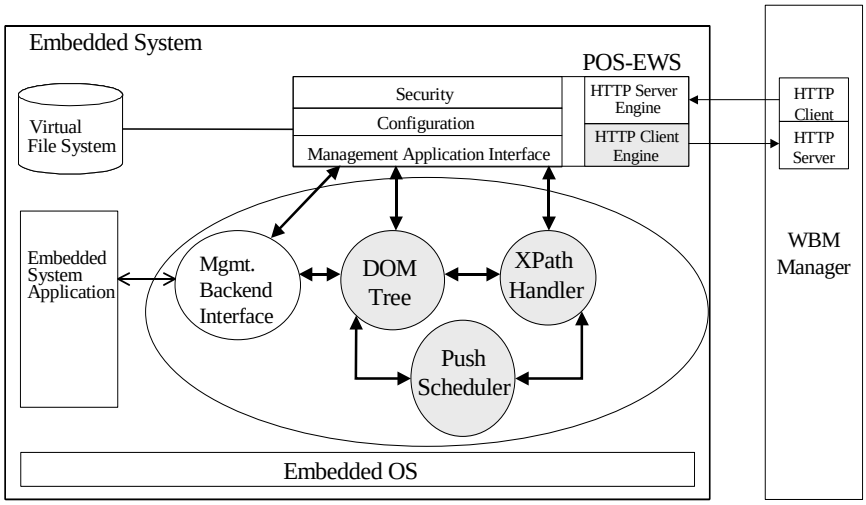


Figure 13. WBM Agent Architecture

The HTTP Client Engine sends asynchronously messages to the WBM manager for reporting alarm and distributing management data according to the schedule. The XPath Handler selects a managed object in the DOM tree, interpreting the XPath expression sent from the WBM manger. A DOM tree is a virtual repository of management data and provide a manipulation point to the managed object. The Push Scheduler collects subscription information and schedule a series of message distribution, and sends the schedule messages.

When the WBM agent receives a request message, the Management Application Module of POS-EWS selects specified nodes in the DOM Tree using the XPath Handler. For selected nodes, it retrieves management data from the DOM Tree through the DOM interface: dump, and send it to the WBM manager. In order to send up-to-date information, the DOM Tree updates the node value for selected node before dump. This type of update is called a pull-based update. For a certain node, it is not necessary to update before dump because its node value is up-to-date information. In this case, the Management Backend Interface module has the responsibility for update. This type of update is called a push-based update. For rapidly ever changing data such as instrumentation, the pull-based update is more appropriate than the push-based update. And static or semi-dynamic information can gain more profit in using the push-based update than the pull-based update. The pull-based update node is implemented by replacing the text of the node value with the Processing Instruction node that is a standard node type of DOM.

When the WBM agent receives a control message, the Management Application Module takes the same procedure in the case of receiving a request message, but executes a registered handler instead of dump information. The Management Backend Interface can send a notification message by means of calling the registered handler at the subject node after a push-based update.

4.2.2 WBM Manager

Figure 14 illustrates the structure of a WBM manager. Web service is

utilized for following two purposes: for providing Web interface to operation and for receipt of asynchronous message from WBM agent over HTTP. Each function is implemented in different URL. Web Client exchanges synchronously management information with WBM agent. The Information Repository is used for storing management information for a long-term analysis. The XML Parser and Translator module provides a basis for implementing most management application functions because management information is represented in XML data. The XML Parser and Translator makes it possible for us to implement management application functions such as filtering, logging into the Information Repository and collecting data from multiple WBM agents.

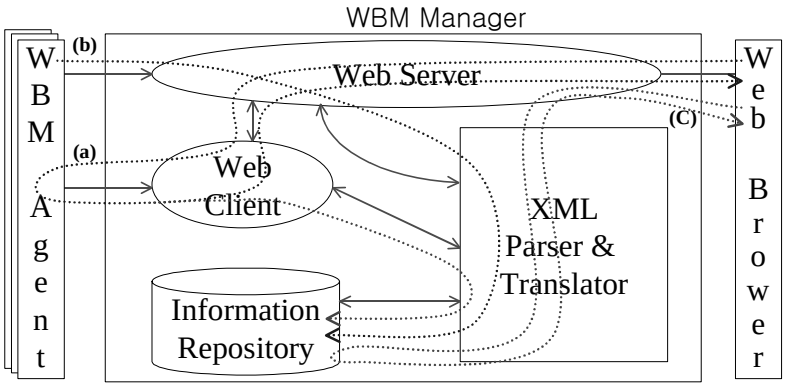


Figure 14. WBM Manager Architecture

There are tree typical information flows within the WBM manager. They are depicted in Figure 14 as dotted arrow line and labeled from (a) to (c) for each flows. The data flow (a) is used for unifying Web-based element user interface into a Web browser interface without any additional management logic. Where the WBM agent send notification of alarm and distribute management information on schedule, the information travels through the data flow (b). The data flow (c) is used for generating report of long-term analysis.

XML schema is used for modeling management information, which is

provided by WBM agent. And it is very closely related with database schema for information repository in WBM manager because generally the structure of collected information is preserved in database of WBM manager. But it is not good approach to use the XML schema in information repository without modification. The first reason is that new information might be added into the collected management information in order to give more significance to the management information. Source and time of gathered information are most typical information to be added when storing. Another reason is that information repository needs new database schema for store additional information that is result of analysis. XML make it easy to implement theses functions in WBM manager.

4.2.3 SNMP Integration

SNMP is the most widely used management method for managing network devices on the Internet. Its simplicity enables it to be implemented on small platforms without much difficulty. Presently, most network devices are equipped with an SNMP agent. With integration of SNMP and Web-based network management architecture, the advantages of Web-based network management architecture are preserved with no loss of SNMP agent functionality.

We regard SNMP agent as a special form of the Management Backend Interface module. The integrated architecture with SNMP agent is depicted in Figure 15. Management information emitted by the module is defined in SNMP MIB. Therefore, XML schema for creating DOM tree in WBM agent can be produced by automatic translation. We have developed a SNMP MIB to XML translator. By replacing each node of DOM Tree for SNMP agent with PI node that issues SNMP messages, a WBM agent can always provide up-to-date SNMP MIB value.

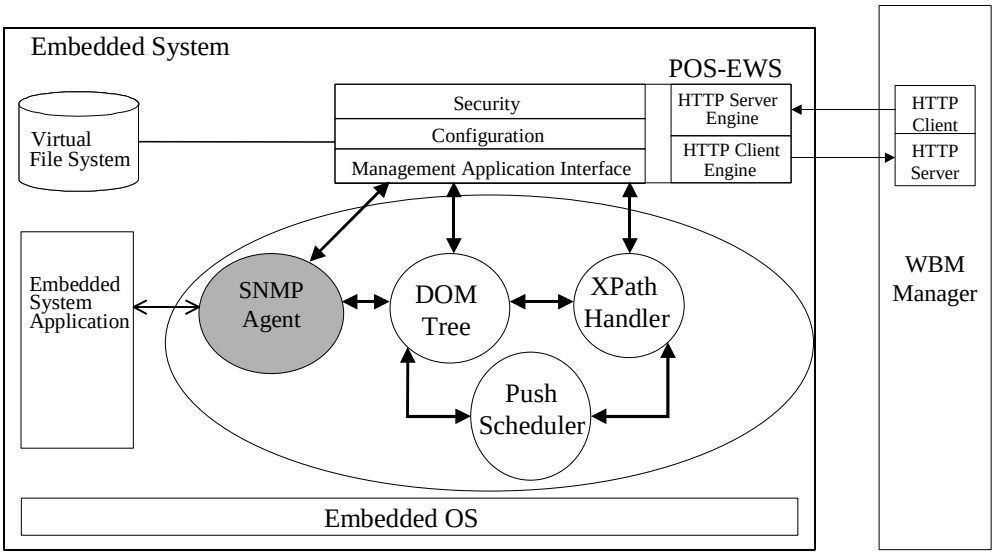


Figure 15. Integrated Architecture with SNMP Agent

4.3 Comparison with previous work

In this section, the proposed Web-based Network Management Architecture (WNMA) is compared with previous work; WBEM and WIMA. Three management architectures uses HTTP/TCP protocol stack for communication, but WBEM defines new management operations and addressing method without support for asynchronous communication. New defined management operation and addressing method are defined in XML schema. WIMA and WNMA are same with HTTP as a management protocol but different to the addressing method for managed object. WNMA uses XPath expression to address managed object, but WIMA does SNMP OID mapping to URI. The dissimilarity of addressing method between two architecture come from the differenc in information model. With respect to information model, three architectures propose different approaches with each other. WIMA do not propose

new information model; uses legacy SNMP SMI. WBEM defines new information modeling approach; new information modeling method (CIM) and mapping CIM to XML. But our proposed architecture uses XML modeling approach; XML schema.

Features	WNMA	WBEM	WIMA
Architecture	Manager-Agent	Clams to support all	Manager-Agent
Information Model	Object-based	Object-oriented	Object-based
Specification Language	XML schema	CIM (MOF, UML, XML)	SNMP SMI
Operations	Use HTTP operation	23 operations	Use HTTP operation
Communication Mode	Sync/Async	Sync	Sync/Async
Addressing	XPath expression	Name and Associations	MIT with OID
Standardization Body	Proprietary	DMTF	Proprietary
Mgmt. Domain	Network Mgmt.	Systems Mgmt.	Unidentified
Protocol Suit	HTTP/TCP	HTTP/TCP	HTTP/TCP

Table 3 Comparisons with previous work

5. Validation

We implement an embedded Web server, POS-EWS that we have developed for Web-based element management and XML-based network management. In this section, we give implementation results of POS-EWS and evaluate POS-EWS's performance in areas such as code size, run-time memory, CPU usage and connection capability. We also explain the methods used in optimizing our POS-EWS. We implement a Web-based element management system for a commercial router for validating the efficiency and effectiveness of Web-based element management architecture.

5.1 POS-EWS

We have implemented an HTTP/1.1 compliant embedded Web server based on the EWS design presented in the previous section. We call this system POS-EWS, which stands for POStech-Embedded Web Server. The C programming language, commonly used in an embedded system, is used throughout the server implementation. We have implemented POS-EWS on the Xinu OS using the MPC 860 processor.

POS-EWS implements a subset of the HTTP features typically required for use in an embedded system. To reduce the TCP connection resources, HTTP/1.1 permits a persistent TCP connection to be established for as long as the Web browser requires access to the server. For providing up-to-date dynamic information, the server needs to control the cache mechanism that is also included in HTTP/1.1. The cache control and persistent TCP connection is essential for an EWS, and POS-EWS supports these two features.

When developing POS-EWS, we approached the problem of supporting

multiple connections in the context of a single thread by implementing the finite state machine, which is introduced in Section 3.2. The implementation result is much efficient than multiple thread approach. The number of states in the finite state machine is fifteen; therefore only 1 byte variable is sufficient to store the state of machine. And the transition table of the finite state machine consists of a function pointer being run at entry point and a variable holding the current state in the finite-state machine. The memory used for implanting state transition table is less than hundred of byte, which is sufficiently small overhead comparing with a thread overhead of a program counter, a register set and a stack space.

POS-EWS uses a Virtual File System (VFS) which can provide a limited set of read-only files built into the ROM. The VFS can be used with or without a real file system. If a real file system exists, the VFS will forward the file request to it. Using the VFS generator, which is one component of the POS-EWS preprocessor compiler, the compressed HTML file for use by the EWS is created. The file will be decompressed by the VFS prior to use by POS-EWS.

POS-EWS also supports state management using HTTP cookies. A cookie is a record that contains management data for a manager to set. It is stored on Web browsers, and is sent to Web servers each time a manager sends a request to a Web server. Cookies are useful for having a Web browser remember some specific information which the Web server can later retrieve. A server, when returning an HTTP object to a client, may also send a piece of state information which the client will store. This simple mechanism can be used in management applications.

We have also developed a Web compiler for constructing a virtual file system (VFS) and efficient SSI application interface. An example of an HTML and a subset of a compilation result are shown in Figure 16. In this example, the content of *sysname.html* is converted into a character array by the name of *sysname_html*, which is the result of simple conversion from file name to C language array name, i.e., changing the dot to an under-score.

The structure *vf* is a container for storing file information such as file size,

last modified date, etc. The pointer value of the converted character array, *data*, is one of the most important elements in the structure because this value is used to read real content by POS-EWS at run time. The structure *sc_list* is used to make a linked list for script functions. The header pointer for the linked list is also one element in the *vf* structure. POS-EWS uses this pointer value for calling the script function. The structure *vf* has an additional variable for supporting the file interface functions, for example, file read pointer, file state flag, etc. With the file interface functions such as file open (*vf_open*), file read (*vf_read*) and file close (*vf_close*), generated C codes become a complete virtual file.

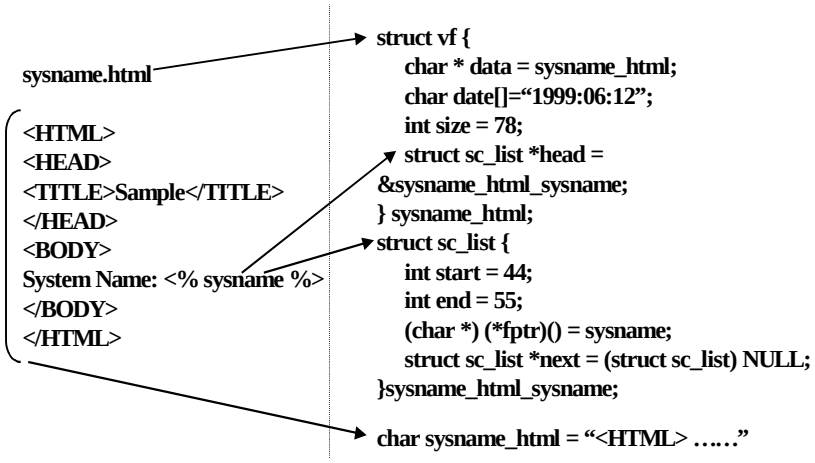


Figure 16. Virtual File System Code

Optionally, the Web compiler can also compress the Web documents. HTML is easily compressed as much as 50% with almost no run time memory required for decompression. HTTP/1.1 supports compressed file transfer from the Web server to Web browser. The Web document is stored in compressed form, transmitted directly, and decompressed by the Web browser. HTTP/1.1 can convey the information of compressed documents in the *Accept-Encoding* and *Content-Encoding* header fields. More importantly, it indicates what

decompression algorithm will be required to remove the compression encoding. The following algorithms, as well as others, are registered in standard HTTP/1.1: *gzip* (generated by the GNU gzip program), and *compress* (produced by the common UNIX file compression program *compress*). Because the algorithms minimize the ROM space used, storing a reasonable size of Web documents on the device has a negligible impact on embedded system resources. For POS-EWS, we have used the *gzip* algorithm to compress at preprocessing and decompress at run time.

The results of implementation can be summarized as follows: the POS-EWS Web compiler converts Web documents to be stored in the virtual file system to compressed C arrays as a virtual file. Then it creates a directory data structure in order to store the file information in the virtual file system. Library functions for the file interface are supported without any RTOS dependency.

5.2 Performance Evaluation of POS-EWS

We developed POS-EWS for Web-based network element management. In this section, we evaluate POS-EWS's performance in areas such as code size, run-time memory, CPU usage and connection capability. We also explain the methods used in optimizing our POS-EWS.

5.2.1 Performance Metrics

The performance of a Web server is dependent upon a number of variables: the server hardware and operating environment, the server application, the network protocol, and the network, hardware, bandwidth and traffic load. Perception of that performance depends also on variables on the client side: the client platform and operating environment, and the Web client.[23] There are four metrics most often used to measure the capacity of general Web servers.

- Requests per second (rps or HTTP ops/sec), which is connections served

or requests made per second.

- Throughput in bytes per second, which is dependent upon the bandwidth of the data pipe.
- Round-trip time (RTT), which is a measure of how long it takes for a packet to be sent from the client plus the time a response from the server takes to be received by the client, completing the request.
- Error rate, which is a measure of how many HTTP requests were lost or not handled by a server.

The two key elements of HTTP performance are latency and throughput.

- Latency is measured by the RTT and is independent of the object size.
- Throughput is a measure of how long it takes to send data, up to the carrying capacity of the data pipe. Improving throughput is simply a matter of employing faster, higher bandwidth networks.

Basically, the performance of a general Web server can be measured by timing how long a server takes to respond to a request and to count the number of bytes delivered per unit time. But EWSs have different performance metrics from general Web servers.

In general, embedded systems must minimize requirements for system resources such as CPU and memory: the embedded Web server is acting as a secondary feature of the system and should avoid interfering with the system's main purpose as much as possible. The most important task of an embedded system is to perform mission-critical and real-time applications. An EWS is often the lowest priority service in the system, so end-users can wait hundreds of milliseconds for a response (an eternity compared to the low-latency requirements of many embedded real-time applications). Therefore, RTT and throughput are not important metrics for an EWS.

The performance element of POS-EWS are code size (memory footprint), run-time memory, the CPU usage and maximum user connectivity (the capacity of

POS-EWS). The code size is approximately 30 Kbytes, the average run-time memory is 64 Kbytes, with the HTTP server having the lowest priority. The POS-EWS supports multiple, simultaneous HTTP transactions and multiple users. By evaluating the system performance we can determine how much an EWS impacts its embedded system. The code size of our POS-EWS is very small so that it puts little impact on the resource scarcity problems.

5.2.2 POS-EWS Optimization

We implemented POS-EWS as a finite state machine (FSM) to improve its performance. We approached the problem of supporting multiple connections in the context of a single process and thread by implementing an FSM, which processes an HTTP request as a sequence of discrete steps. The “process-per-connection” architecture, which forks a new process for each connection, while it goes by the stateless model of the HTTP scheme, is less than efficient. The time and resources required by the fork and exec operations are significant, particularly in light of the fact that a typical Web request is very short. But the FSM supporting single thread is run by a small scheduling system that uses lightweight task structures. This makes the CPU usage and the memory footprint reasonable.

We also give our HTTP engine more improved performance following the HTTP standard. We implemented POS-EWS to keep TCP connections open and reuse them by the Keep-Alive option. Therefore, the cost of opening a new connection for each transaction can be eliminated by reusing existing TCP connections. In this way the transaction time will be the time to send a request plus the RTT plus the processing time on the server plus the time to send the response. When the Web server keeps TCP connections open and does not close them at end of an HTTP exchange, the maximum number of available TCP connections will be reached. The Web server then closes the oldest idle connection first. HTTP version 1.1 specifies the optional use of the Keep-Alive connection. Requesting a Keep-Alive connection when GETing a file means the browser can reuse the connection to get subsequent files from the server.

The POS-EWS Web compiler preprocesses and compresses Web pages and images into compilable ANSI-C code. This allows pages to be developed using standard HTML tools, and stored internally in an efficient format. The Web compiler makes it possible to minimize the application memory footprints through intelligent compression. Shared and nested pointer techniques are used for additional memory savings. The Web compiler reduces the processing time through preprocessing.

We used APIs to extend a server's functionality. Instead of having to parse the incoming amorphous stream of form data, POS-EWS uses options for receiving it, often preformatted and type converted into C struct fields, ready for program use. Using an API instead of a CGI has the advantage of integrating the extensions to the server within the server process. This eliminates the need to go to the OS for communications between the server and the script. Such a C level interface can save much time and simplify CGI programming immensely.

5.2.3 Comparison with other embedded Web servers

In this section, we briefly investigate embedded Web server products, focusing on their features. Web servers can have a range of capabilities and still be http-compatible. A number of commercial EWS products have appeared on the market, each with its own particular value position. There are also many computer communities exploring small Web servers. They make a Web server the size of a matchbox. The Matchbox Server [54]is a single-board computer with a 16 MB RAM, and 16 MB flash ROM, big enough to hold a useful amount of Linux including the HTTP daemon. In addition, the iPic Web server [55] is the smallest Web server. It is about the size of a mere match-head. The single chip computer runs the iPic web-server, the world's tiniest implementation of a TCP/IP stack and an HTTP web-server.

Table 3 and 4 list and compare the features of a number of commercially available Embedded Web Server implementations and our POS-EWS. Blank columns represent features not supported or we could not find appropriate

information on them from the available literature. We summarize the offerings available and the approximate code size needed. This range does not necessarily reflect differences in code efficiency, however. Most EWSs offer small footprints lower than 100 Kbytes for low resource utility, dynamic content generation of SSI type mechanism, some kind of page compression, and options for security/authentication and porting layers to accommodate custom file system and TCP/IP stack.

All of the above products work with any standard browser, including Internet Explorer, Netscape Navigator or Communicator, Hot Java and Mosaic, on all platforms, allowing any network-connected browser user to easily access any device. As well, all serve multiple, simultaneous users. Processing multiple requests may be important if more than one user is to access the embedded system at the same time, or if the system is to report itself busy to potential users. A few support a Web compiler and Virtual File System (VFS), which are essential features for enhancing Web ability and efficiency. Only RomPager and our POS-EWS clearly specify HTTP cookies for state management. POS-EWS has full features of EWS to support the functionality of EWS.

Also, our POS-EWS provides effective integration mechanisms into embedded management applications. POS-EWS offers four basic interface mechanisms for use between a management application and application of an embedded system and an embedded Web server or Web documents. A developer can choose an appropriate interface mechanism depending on the characteristics of management information or types of Web documents. We testified them through applying POS-EWS to management of a commercial router. The integration mechanisms into embedded management applications are important when an EWS is applied to network element management

Company& Product	OS supported	CPU supported	HTTP code size (version)
Agranat Systems, EmWeb	No OS	Any CPU with a C compiler	25kbytes(1.1)
AllegroSoft, RomPager	Any RTOS, No OS	Any CPU with a C compiler	10-40 kbytes(1.1)
BVM IntraScada, Web Server	OS-9	CPU32	< 100kbytes(1.1)
Accelerated Technology, Nucleus Embedded Software	Nucleus Plus	x86,68K,ARM, 683xx,SPARC, PowerPC,SH,	40kbytes(1.0)
Spyglass, MicroServer	LynxOS, QNX, pSOS, OS-9, VxWorks	Any CPU with a C compiler	35-110 kbytes(1.1)
QNX Software Systems Ltd QNX Internet Toolkit	QNX real-time OS	x86, Pentium Pro, AMD	106 kbytes(1.1)
Magma, Lava	Any RTOS	Any CPU with a C compiler	15-40 kbytes(1.0)
Quiotix QEWS	pSOS, LynxOS, VxWorks	Any CPU with a C compiler	45-50 kbytes(1.0)
Web Device, Pico Server	LynxOS, Nucleus Plus, pSOS, VxWorks	Any CPU with a C compiler	15-30 kbytes(1.0)
POSTECH, POS-EWS	Real-time Xinu, pSOS	Any CPU with a C compiler	30kbytes(1.1)

Table 4. Embedded Web Server Products

Company& Product	SSI	VFS	Compiler	Compression	Security(encode)	Cookie
Agranat Systems, EmWeb	O	O	O	Proprietary	Basic +Digest	
AllegroSoft, RomPager	O	O.	O	Dictionary	Basic	O
BVM IntraScada, Web Server				Unspecified	Basic	
Accelerated Technology, Nucleus Embedded Software	O			Propritary	Basic(DES)	
Spyglass, MicroServer	O			None	Baisc + Digest + SSL	
QNX Software Systems Ltd, QNX Internet Toolkit	O				Basic + Digest(encode)	
Magma, Lava	O			Proprietary	Basic + SSL	
Quiotix, QEWS	O	O		GZIP	Basic	
Web Device, Pico Server	O	O		ZIP-like	Basic + Digest + SSL	
POSTECH, POS-EWS	O	O	O	GZIP CSS-Style	Basic + Digest(base64)	O

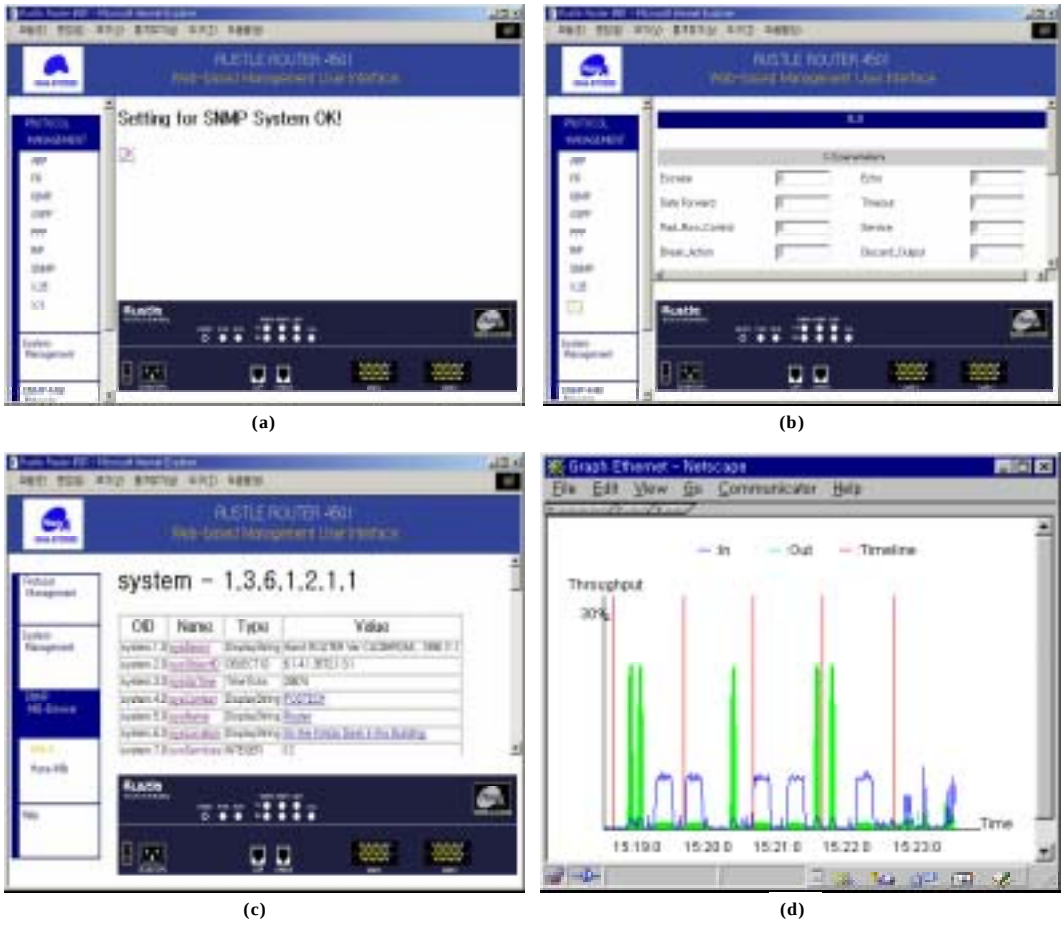
Table 5. Feature comparison of embedded Web servers

5.3 Validation of Web-based Element Management Architecture

We developed a Web-based element management user interface for a commercial Internet router (Hana Systems' Rustle router 4501 [56]) that had been equipped only with a CLI interface and SNMP agent. At first, we developed and embedded POS-EWS into the router and mapped the CLI interface into a Web user interface. All CLI interfaces were classified into two classes: command processing and state reporting. For the command processing class, we applied the CGI-Type interface mechanism; and for the state reporting class, we used the SSI-Type interface mechanism. The CLI interface management application program was converted to the Web interface management application program for exclusive use by POS-EWS. The text output functions of the CLI interface were converted to formatted write functions using a CGI library provided by POS-EWS, and the input text parsing functions of the CLI interface were converted to Web parameter parsing functions using a CGI library.

By the virtue of the Web compiler, Web interface design for the SSI style was completely separated from management application development. Bruins's work [57] persuaded us not to map commands to URLs. Each CLI command was mapped to menu item and HTML links in Web pages. After converting the CLI interface to Web-based user interface, we added an SNMP MIB browser to Web-based user interface without additional programming by use of MIB2HTML compiler and SNMP agent interface library. Only the memory footprint of the code was increased to the extent of storing the generated HTML file. When SNMP MIB-II and a 40-variable private MIB was added, the memory footprint was increased by 40 kbytes. After adding MIB browser, we could perform all management functions to the router through Web-based user interface. But a real-time display for traffic monitoring and alert action on notification was not

supported, So a Java SNMP manager library was used to compensate for that shortcoming.



(a) (b)

(c) (d)

Figure 17. Example of POS-EWS Web-based user interface

Figure 17 shows the developed Web-based user interface examples. Figure 17 (a) and (b) are a converted Web-based user interface from CLI by CGI-Type and SSI-Type interface mechanism, respectively, (c) is an example of Web-based user interface for the MIB browser and (d) shows a real-time traffic monitoring example. Only in embedded environment, we implemented Web-based user

interface and validated the effectiveness of its four interfaces. In general-purpose computer environment, you must consider other interface mechanisms because you can use Java Virtual Machine, advanced shell tools and advance script languages, etc.

5.4 Validation of Web-based Network Management Architecture

We are developing a global management system for commercial high dense Linux server (Netstech EnterFLEX series [58]) based on the Web-based network management architecture. In EnterFLEX, tens of single board computer (SBC) are integrated into single chassis and each SBC can be used as a standalone, autonomous server or the cluster. The EnterFLEX supported only Web-based element management for each SBCs, called Server Blade Manager (SBM).

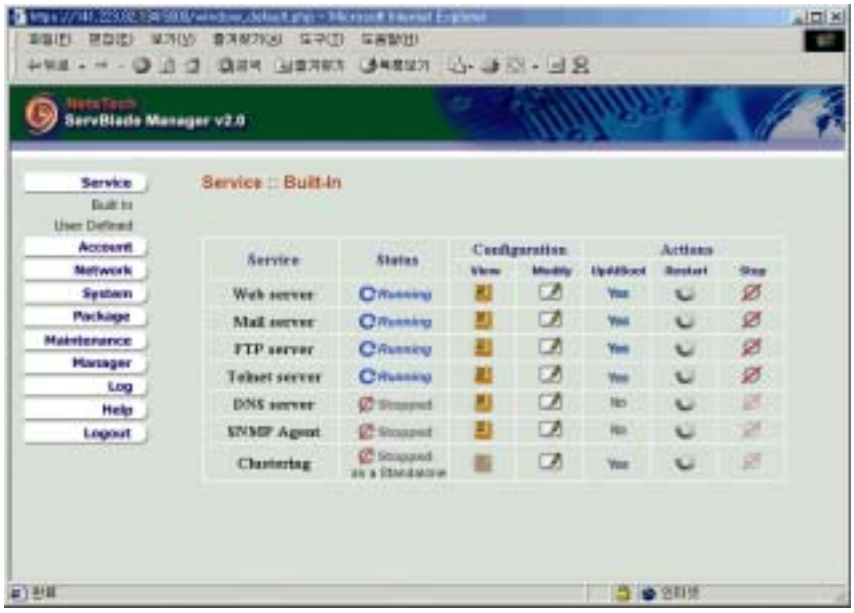


Figure 18. Sample ServBlade Manager User Interface

Figure 18 shows a sample SBM Web-based user interface that displays the state of well-known services such as Web, Ftp, etc. An operator must open a Web browser for each SBCs in order to use a SBM. When an operator is in charge of tens or hundred of SBC, it is nearly impossible to monitor the state of the assigned SBCs because the number of opened Web browser and monitored SBC are always the same. Further, an operator must spend a lot of time or effort on open and close of Web browser if large number of SBC need to be reconfigured.

The Global Server Blade Manager (GSBM) system was developed for eliminating above inconvenience and inefficiency. The GSBM gives an operator a Web-based management user interface for hundred or thousand of Linux server through a Web browser. At first, we extended the POS-EWS to support XML. The applied XML package for extending was libxml2 [59], which is an open source C language implementation and supports DOM and XPath.

We worked out the management information of GSBM based on Web display of SBM. Management information model of GSBM in XML DTD are depicted in Figure 19. The rules for deriving management information model from Web display are as follows

- The root element of GSBM DTD is “ServeBladeManagers”.
- All items in menu, hyperlink and table header are defined DTD elements, while preserving their hierarchy.
- All items in action button (e.g. “add”, “delete”) are defined as XML attributes of target element under the same attribute name of “operation”.

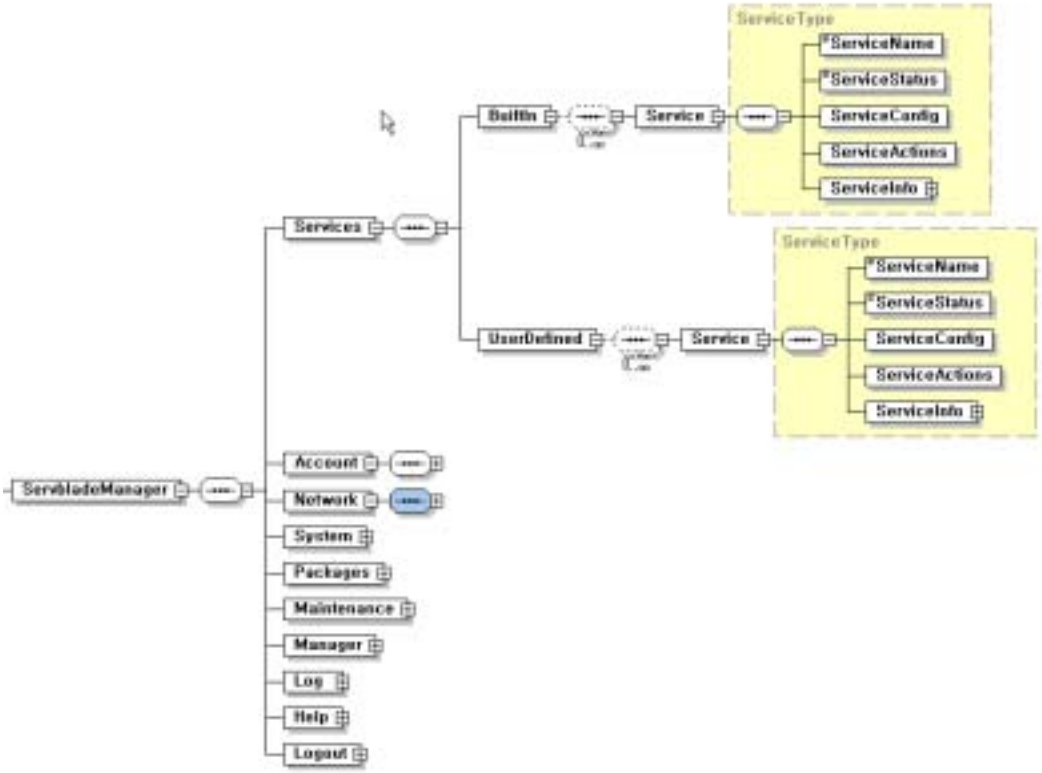


Figure 19. Management Information Model of GSBM

We implemented a WBM agent for GSBM based on derived XML DTD. We reused the SBM backend program, which provide real resource interface. And for generating and parsing the XML document that valid for GSBM DTD, we only configure the XML parser and generator of WBM agent platform. In consequence, we developed a WBM agent of GSBM with a little modification of backend program and platform configuration, without source programming.

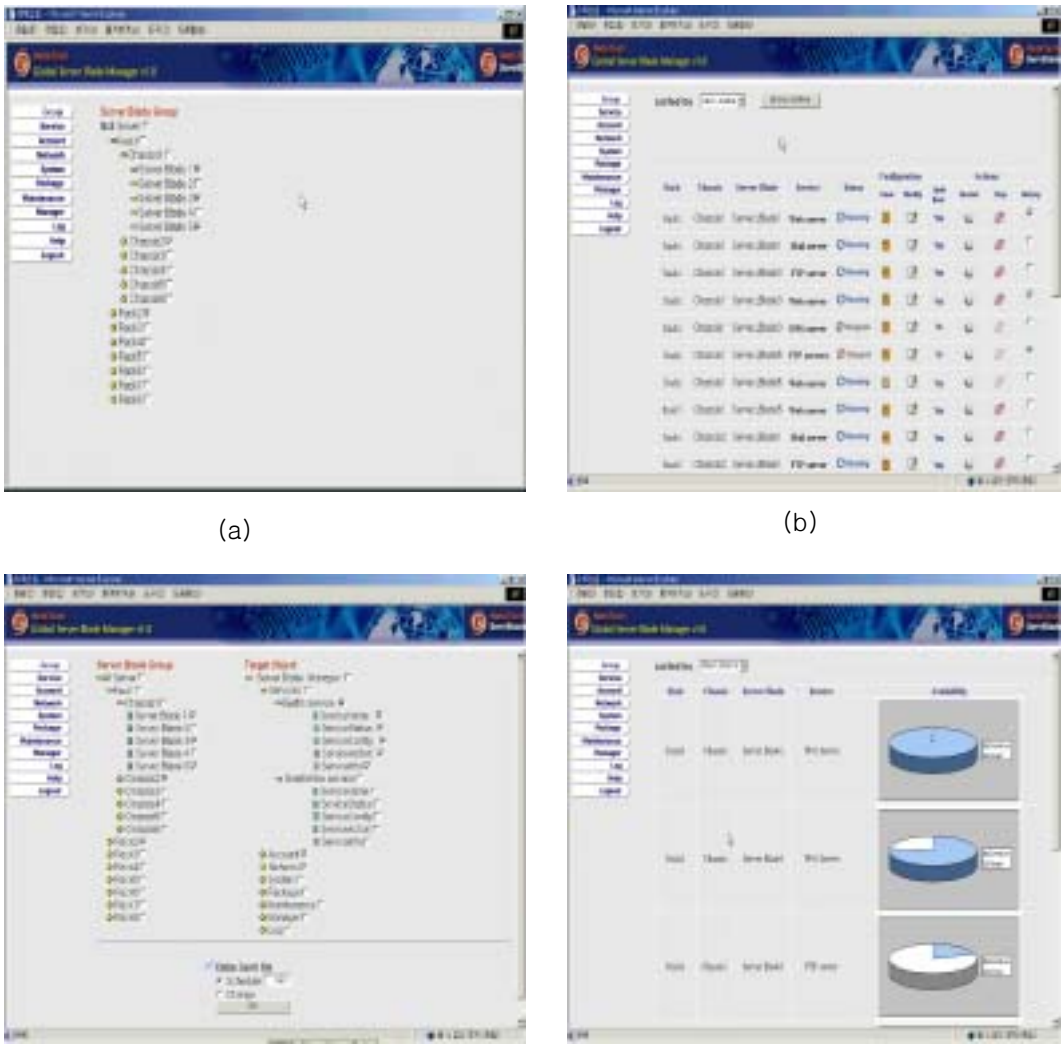
We implemented a WBM manager for GSBM. It basically supports all management functions that were provided by SBM. Undoubtedly SBC grouping mechanism was introduced in the WBM manager in order to handle multiple SBC simultaneously. The network management functions as well as the above interface

unification were added to the WBM manager. The added functions are as follows:

- Logging for service availability
- Trend analysis for CPU, disk and memory usage.
- Alarm notification for server fail.
- Report generation.

These functions require continuous state monitoring for managed resources and information repository for long-term analysis. The efficient use of bandwidth for monitoring is very important factor because number of managed system is very large. This lead us to adopt a push-based organization model. Monitoring data for long-term analysis are pushed by WBM agent, not pulled by WBM manager. Information repository was implemented using database management system. We can save much time and cost by virtue of broad range of XML support from database management system and commercial off-the-shelf components.

Figure 20 shows examples of GSBM user interface. User interfaces of Figure 20 (a) and (b) give an operator a unified management interface for multiple SBCs. An operator selects SBCs through the user interface of Figure 20 (a). Thereafter management information from selected SBC is collected and displayed on a Web browser. An operator requests information delivery on schedule to GSBM agent through the user interfaces of Figure 20 (c). And Figure 20 (d) show the display of analysis result for collected and stored information, which were subscribed through the user interface (c) by an operator and logged into information repository by GSBM manager.



(a) (b)

(c) (d)

Figure 20. Example of GSBM user interface

6. Conclusion and Future work

In this last section, we summarize the main contribution of our work and give some directions for future work.

6.1 Summary

Web servers are already being built into many network devices today. In the near future, we can expect this trend to grow even further to home appliances, medical instruments and industrial equipment. Embedded web server is an enabling technology in stimulating this trend. Embedded Web servers for Web-based network element management provide an administrator with a simple but enhanced and more powerful user interface without additional hardware. Without the management application specific aspect of embedded Web server technology, these advantages may disappear. We presented our design and implementation of an HTTP/1.1 compliant embedded Web server (called POS-EWS). POS-EWS supports all essential Web server functions to be used for management application in embedded systems. These functions include persistent connection, cache control, cookie mechanism, security and application interface mechanisms. Also it has good performance in memory and CPU usage by applying optimization technique. Implementing by finite state machine, compressed virtual file system and preprocessing for Web documents are applied optimization techniques.

We designed an Web-based element management architecture. The architecture provides an easy but powerful integration platform with Web documents and embedded management application. This feature enables developers to put a Web-based element management interface into the device effectively and efficiently. Web-based element management architecture has four

interface mechanisms with their own unique advantages. Effective integration mechanisms were also introduced for interface mechanisms. We demonstrated effective use of these application interfaces and integration mechanisms using POS-EWS in a commercial Internet router.

This thesis provides a unified embedded Web server architecture that can be applied for network management as well as element management. XML plays an important role as an enabling technology in providing solutions to extend usage of embedded Web server to the network management. It is used for management information modeling and manger-agent communication. XML schema and XPath is used for the management information specification and addressing mechanism. Management information is encoded in XML data and transferred over HTTP. We have applied XML DOM to uniform representation and unified the access mechanism of management data in a WBM agent. By the appropriate and effective use of XML technologies, we have maximized the advantages of using XML in network management.

6.2 Contributions

The core contributions of this Ph.D. work are three fold. First, We design and implement an embedded Web server, POS-EWS, which is suitable for Web-based management. Second, we design Web-based element management architecture that makes it possible for developers to implement a powerful Web-based element management system very efficiently by providing compilers and libraries for real-time and dynamic update, as well as simple interface. Third, we extend the Web-based element management architecture to the Web-based network management architecture based on XML.

POS-EWS uses such a little computing resources that it can be embedded into a small network device, while supporting necessary functions for implementing Web interface. Further, it can be ported into industrial equipment,

home appliance and office automation machines if network interface is added. The optimization techniques being applied for developing the POS-EWS have a wide application such as a small Web browser, mail server, etc.

We defined four interfaces between Web document and embedded management application. The interfaces cover all necessary interactions between them for generating various types of Web documents from static to dynamic and real-time. We present developers with a selection guideline of interface based on analysis results of the interface features. Each defined interfaces has an integration mechanism so as to minimize the developer's efforts. The integration mechanism makes it possible to simplify the development procedure by use of compilers and libraries. Especially for the case of SNMP agent already equipped, Web-based element management system can be produced without a line of source programming. The Web interface supports real-time display and trap notification as well as MIB browsing functions.

Web-based network management architecture gives a way to collect and aggregate management information provided by the Web-based element management interface. The Web-based network management architecture incorporates XML technology so as to develop management application efficiently and manage network effectively by developing the advantages of XML technology.

XML-based network management architecture save the overhead of embedded system by reuse of embedded Web server to support network management interface, replacing SNMP agent with embedded Web server. Also advanced feature of Web technology will solves the problems in SNMP-based management.

WBEM looks a lot like the perfect revolution of network and system management. But it primarily concentrates on systems management. It is not certain that the solution from WBEM is appropriate to the IP-based network management, especially for embedded systems. We expect the result of our research feed into WBEM so as to give a better solution than current one and

enlarge their application area.

6.3 Future Work

We have a plan to study how to integrate other organization models such as policy-driven management, management by delegation, mobile and intelligent agent, etc. into our architecture. Many research work already prove the effectiveness of these organization models in network management area. Preliminary work suggests that these models can be easily incorporated into Web-based network management architecture by virtue of its high flexibility and openness. Adopted Web technologies in our architecture are so flexible that they have a broad application area. And degree of openness of Web technology is preserved in our architecture because we do not introduce new management domain specific concepts into our architecture.

The work presented herein focuses on element and network management. Another worthwhile research area is the suitability of higher management layer: service and business management. Also the suitability check for other management domains such as QoS management, application management, telecommunication and mobile network is another interesting future research topic.

In section 4.2.3, we explained about SNMP integration with Web-based network management architecture where SNMP agent is already equipped into managed system. Another important research issue for integrating with SNMP is how to make SNMP manager manages WBM agent.

References

[1] ITU-T Recommendation X.700, “Management Framework for Open System Interconnection (ISO) for CCITT applications”, September 1992.

[2] ITU-T Recommendation M.3000, “Overview of TMN Recommendations”, October 1994.

[3] ISO 9595: “Information Processing System – Open System Interconnection – Common Management Information Service Definition”, Geneva, 1990.

[4] ISO 9596: “Information Processing System – Open System Interconnection – Common Management Information Protocol”, Geneva, 1991.

[5] ISO 7498-4: “Information Processing System – Open System Interconnection – Basic Reference Model – Part 4: Management Framework”, Geneva, 1989.

[6] J. Case, M. Fedor, M. Schoffstall and C. Davin, “The Simple Network Management Protocol (SNMP),” RFC 1157, May 1990.

[7] WBEM, “WBEM Initiative,” http://www.dmtf.org/wbem/.

[8] JMX, “Java Management Extension (JMX) Home Page”, http://java.sun.com/products/JavaManagement/.

[9] D. Raggett, A. Le Hors, I. Jacobs, “HTML 4.01 Specification,” IETF HTML WG, http://www.w3.org/TR/html401, Dec. 24 1999.

[10] R. Fielding, J. Gettys, J. Mogul, H. Frystyk Nielsen, L. Masinter, P. Leach and T. Berners-Lee, “Hypertext Transfer Protocol - HTTP/1.1,” RFC 2616 IETF HTTP WG, June 1999.

[11] Sun Microsystems Inc., Java Management Programmer’s Guide, Developer’s Release, June 1996.

[12] W3C, “Extensible Markup Language (XML)”, http://www.w3.org/XML.

[13] McCombie, B., “Embedded Web servers now and in the future,” Real-Time Magazine, no.1, March 1998, pp. 82-83.

[14] Ian Agranat, “Embedded Web Servers in Network Devices,”

Communication Systems Design, March 1998, pp. 30-36.

[15] H. T. Ju and J. W. Hong, “POS-EWS Web Compiler for Supporting An Effective Application Interface,” PIRL-TR-99-003 Technical Report, POSTECH, Korea, Dec. 1999.

[16] R. Fielding, “Hypertext Transfer Protocol - HTTP/1.0,” RFC 1945, IETF HTTP WG, May 1996.

[17] Internet Engineering Task Force (IETF), http://www.ietf.org.

[18] W. Stallings. “SSL: Foundation for Web Security”. The Internet Protocol Journal, 1(1):20–29, 1998.

[19] CGI, http://www.w3c.org/cgi.

[20] BMC Software, Cisco, Compaq, Intel, and Microsoft. “Industry Leaders Propose Web-Based Enterprise Management Standards Effort”. Press Release, July 1996. http://www.microsoft.com/mscorp/presspass/press/1996/jul96/WEBMAN~1.htm.

[21] R. Booth. “XML As a Representation for Management Information—A White Paper” Draft. Microsoft, May 1998.

[22] DMTF, “Common Information Model (CIM) Specification Version 2.2”, June, 1999.

[23] DMTF, “Working Groups”, http://www.dmtf.org/member/workgroup.php.

[24] H. Nielsen, P. Leach, and S. Lawrence. “An HTTP Extension Framework”. IETF, RFC 2774, February 2000.

[25] Sun, “Java Management Extensions Instrumentation and Agent Specification, v1.0”, July 2000.

[26] W3 Consortium, “Resource Description Framework (RDF) Schema Specification 1.0”, W3 Consortium Candidate Recommendation, March 2000.

[27] W3 Consortium, “XML Schema Part 0,1,2”, W3 Consortium Recommendation, May 2001.

[28] OMG, “XML Metadata Exchange Version 1.1”, October 1999.

[29] W3 Consortium, “Document Object Model (DOM) Level 1 Specification”,

W3 Consortium Recommendation, October 1998.

[30] W3 Consortium, “Extensible Stylesheet Language (XSL) Version 1.0”, W3 Consortium Candidate Recommendation, November 2000.

[31] W3 Consortium, “XSL Transformations Version 1.0”, W3 Consortium Recommendation, November 1999.

[32] J.P. Martin-Flatin. "Web-Based Management of IP Networks and Systems". Ph.D. thesis, Swiss Fed. Inst. of Technology, Lausanne (EPFL), October 2000.

[33] A. John, K. Vanderveen and B. Sugla, “XNAMI-An extensible XML-based paradigm for network and application management instrumentation”, Proceedings of IEEE International Conference on Network, 1999. pp. 115 – 124.

[34] C. Ensel and A. Keller, “Management Application Service Dependencies with XML and the Resource Description Framework”, Proc. of the 2001 IFIP/IEEE International Symposium on Integrated Network Management (IM 2001), May 2001, pp. 661-674.

[35] H. T. Ju and J. W. Hong, “MIB2HTML Compiler,” PIRL Technical Report, PIRL-TR-99-002, POSTECH, Korea, Dec. 1999.

[36] Hong-Taek Ju, J. Won-Ki Hong, "POS-EWS Web Compiler for Supporting An Effective Application Interface," PIRL Technical Report, PIRL-TR-99-003, December 1999.

[37] Hong-Taek Ju, Mi-Joung Choi and James, W. Hong, “EWS-Based Management Application Interface and Integration Mechanisms for Web-Based Element Management”, Journal of Network and Systems Management, Vol. 9, No. 1, pp. 31-50, 2001.

[38] M. J. Choi, H. T. Ju, H. J. Cha, S. H. Kim and J. W. Hong, “An Efficient and Lightweight Embedded Web Server for Web-based Network Element Management,” in Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS 2000), Hawaii, April 2000, pp. 187-200.

[39] Hong-Taek Ju, Mi-Joung Choi and James W. Hong, "An efficient and

lightweight embedded Web server for Web-based network element management," International Journal of Network Management, Vol. 10, Issue 5, September/October 2000, pp. 261-275.

[40] K. Arnold and J. Gosling, The Java Programming Language, Addison-Wesley, 1996.

[41] T. Berners-Lee, L. Masinter and M. McCahill, “Uniform Resource Locators (URL),” RFC1738, Dec. 1994.

[42] The World Wide Web Security FAQ, http://www.w3c.org/Security/faq.

[43] Stanford University, Matchbox Server, http://wearables.stanford.edu/.

[44] University of Massachusetts, iPic Web server, http://www-ccs.cs.umass.edu/~shri/iPic.html/.

[45] B. Phillips, “Designers: The browser war casualties,” IEEE Computer, Vol. 31, Issue 10, Oct. 1998, pp 14-16,21.

[46] ISO 8879, “Information processing -- Text and office systems -- Standard Generalized Markup Language (SGML)”, Aug. 2001

[47] Tim Bray, Jean Paoli and C. M. Sperberg-McQueen, “Extensible Markup Language (XML) 1.0”, W3 Recommendation REC-xml-19980210, February 1998.

[48] W3 Consortium, “XML Path Language (XPath) Version 1.0”, W3 Recommendation, W3 Consortium, November 1999.

[49] T. Berners-Lee, R. Fielding, and L. Masinter, “Uniform Resource Identifiers (URI): Generic Syntax”. IETF RFC 2396, August 1998.

[50] W3 Consortium, “WWW Security FAQ: Running a Secure Server”, http://www.w3.org/Security/faq/wwwsf3.html.

[51] Netscape, “An Exploration of Dynamic Documents,” http://home.mcom.com/assist/net_sites/pushpull.html

[52] Li Gong, “Java 2 Platform Security Architecture,” http://java.sun.com/j2se/1.4/docs/guide/security/spec/security-spec.doc.html

[53] Castedo Ellerman, “Channel Definition Format (CDF),” W3 Consortium Recommendation, Nov. 2000.

[54] Stanford University, “Matchbox Server,” http://wearables.stanford.edu.
[55] University of Massachusetts, “iPic Web Server,” http://www-ccs.cs.umass.edu/~shri/iPic.html.
[56] Hana Systems Inc., http://www.hanasys.co.kr.
[57] Barry Bruins, “Some Experience with Emerging Management Technologies,” The Simple Times, vol. 4, No. 3, July, 1996, pp. 6-8.
[58] Netstech Inc., http://www.netstech.com
[59] The XML C library for Gnome, http://www.xmlsoft.org