Efficient Enterprise Integration Solutions Using Web Services for Hospital Management System

Savitri Bevinakoppa Kaushik Hegde
School of Computer Science and Information Technology, RMIT University
Melbourne – 3001, AUSTRALIA
savitri@cs.rmit.edu.au khegde@cs.rmit.edu.au

Abstract

Distributed oriented businesses are moving towards mergers and tie-ups with other businesses, so as to provide value to their customers and thereby increase their revenue etc. They are facing problems of efficiency in integrating their heterogeneous software applications with each other. The evolution of Web services has simplified the problem of Integrating Enterprise (IE) applications. Web services use technologies, which are Simple Object Access Protocol (SOAP), used for messaging, Web Services Description Language (WSDL) used for description and Universal Description, Discovery and Integration (UDDI) used for discovery. These technologies improve transactional integration and add value to the business workflow. However the core technologies involved in Web services like SOAP are riddled with problems of performance and slow response time.

This paper gives the development of a Hospital Management system, which is dynamically integrated to external and internal online hospital services. The services include patient registry, pathology, radiology and radiotherapy, using web services with SOAP, WSDL and UDDI. The performance and response times are improved by caching SOAP requests and responses. From the experimental results, it was found that caching SOAP messages helps in improving the performance of the web services infrastructure. This research establishes the fact that web services can be effectively used for integrating applications dynamically.

1. Introduction

Web Services is a new revolution in the field of enterprise applications having program-to-program interactions such as Business-2-Business portals [1]. It reduces the costs for business moving towards e-business help in deploying applications very fast and dynamically discovering new opportunities. These advantages resulting from web services are due to the fact, that they are built on commonly used and emerging standards like HTTP, Extensible Markup Language (XML) [2], Web Services Description Language (WSDL) [3] and Universal Description, Discovery and Integration (UDDI) [3]. It allows applications to be integrated economically, fast and easily because integration of business is not based on network protocol semantics, but on messages generated based on service semantics, thus inducing loose coupling between business applications [4]. It provides a uniform approach to integrate external applications as well as internal applications and evolve electronic business towards just in time integration of business over the Internet.

The objective of this research is to build efficient web-based Hospital Management system. This solution enables doctors to interact with various internal and external services that are dynamically integrated using the concepts of web services. The various services may include patient details service, pathology service, radiology service and radiotherapy service. A service provider such as the radiology department registers a service with the UDDI registry of hospital management. The service functionality is exposed using a WSDL file, which is registered in the hospital management UDDI registry. The performance of transmission of request and response SOAP messages between the service requestor (Hospital Management solution) and the service provider (radiology etc) is improved by caching these SOAP request and responses.

2. Web Services

The web services model consists of three main roles namely service provider, service registry and service requestor and three main operations namely publish, find and bind. The conceptual web service stack is made up of 5 layers namely service discovery, service publication, service description, XML messaging and network. These layers are guided or have to satisfy the factors of security, management and quality of service.

Service Discovery and Service Publication layers of the web service stack are involved with service discovery and service publication functionality of the web service model and they are implemented using Universal Description, Discovery and Integration (UDDI). It is also responsible for describing the service operation and also gives binding information. It is implemented using web services Description Language (WSDL).

XML Messaging and Network layers of the web service stack are responsible for transportation of request and responses between service initiator and service provider. The XML messaging is implemented using Simple Object Access protocol (SOAP), which is transported over HTTP or SMTP or FTP in the network layer [5].

2.1 Simple Object Access protocol (SOAP)

Simple Object Access protocol is a standardized XML based object invocation protocol [5]. SOAP [6] was initially developed so that distributed applications could communicate with each other over HTTP which is one of the standard ways to communicate over the internet and through corporate software security systems like firewalls, other distributed protocols like RMI, IIOP and COM were ineffective in these regards. Since SOAP is defined in XML and is based over HTTP it is platform independent. SOAP does not define its own application semantics in terms of programming model and implementation specific semantics rather it defines a simple mechanism for expressing application semantics by providing a modular packaging model and encoding mechanisms for encoding data within modules. This enables SOAP to be used in a variety of applications ranging from messaging and remote procedure calls (RPC).

3. Hospital Management System

A Hospital Management system has been taken as a case study in this paper. The solution provided for a Hospital Management system proves that web services are an efficient integration solution. The Hospital Management solution shows how caching improves the performance of SOAP.

3.1 Hospital Management Solution Description

A Hospital Management solution will provide a revolutionary single point of access, to many external and internal Hospital services like radiology service, pathology service, hospital management service etc, to its clients mainly the doctors. The doctor will be able to access patient details such as patient personal information, diagnostic information etc belonging to various hospitals from a single access point. The Hospital Management solution will be making use of functionalities of already existing software solutions of various departments like radiology department, pathology department etc, which may be under different hospitals to achieve its functionality. An integration framework has to be applied to the Hospital Management solution for its successful interaction with heterogeneous software applications of various departments. In this minor project a web services integration framework is selected for the Hospital Management solution due to the advantages of web services over other existing integration solutions.

3.2 Main Functionalities of Hospital Management Solution

The doctor can view patient contact details of a patient belonging to various hospitals that have exposed their services as a web service to the Hospital Management system. The doctor records patient clinical exam details using the Hospital Management system.

3.2.1 Stakeholders involved in the Hospital Management system

Doctor: -A doctor uses the Hospital Management system to view and record patient contact details and diagnostic details.

The doctor uses the Hospital Management system to record patient clinical exam details.

Radiology Department: uses the Hospital Management system to expose its services such as treatment and its outcome.

Radiotherapy Department: uses the system to expose its services such as dosage, location etc.

Pathology Department: uses the system to expose its services such as MRI, PET images.

Toxic/Follow-up Department: uses the system to expose its services such as consultation and its outcome. Architectural view of Hospital Management system is shown in the figure 1.

3.3 Deployment view of Hospital Management system

SOAP cache is used to cache SOAP requests and responses. UDDI registry is used for registration and dynamic discovery of services. The deployment view of Hospital Management System is shown in figure 2.

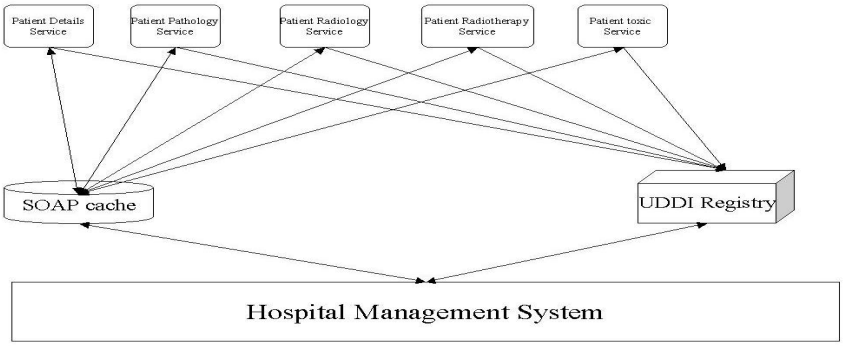

Figure 1 Architectural View of Hospital Management System

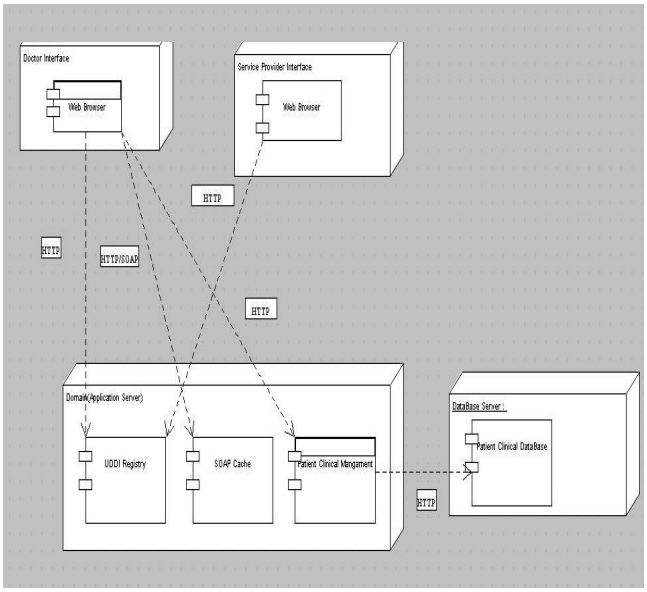

Figure 2 A Deployment View of the Hospital Management System

The top level indicates the client view of the doctor and the service provider. The client view interacts with the application through the web browser. The domain layer hosts the UDDI registry, SOAP cache and clinical management component of the Hospital Management system. The data server layer is used to host the database used to store patient clinical details. The protocol used for transmission between the layers is HTTP coupled with SOAP.

4. SOAP Cache

In Distributed software applications using a web services infrastructure, the default protocol used for transmission of requests and responses is Simple Object Access Protocol (SOAP). The performance of SOAP compared to other Distributed binary protocols like RMI, IIOP and COM is very poor. As SOAP is an XML based protocol, assembling and disassembling a SOAP request and response involves a lot of overhead. Caching a SOAP request and its corresponding response locally on the client end helps in improving the performance of SOAP bypassing the time required to assemble and disassemble a SOAP message. Only read SOAP request are cached as caching of write SOAP request leads to inconsistency and failure of the entire software application. The following section gives details of the architecture of caching component, design details involved in eviction of SOAP objects from the cache and maintaining consistency of SOAP messages in the cache, the implementation of the SOAP cache and the test results regarding the performance of SOAP by caching SOAP requests and responses locally at the client end [7], [8].

4.1 The Main Components of SOAP Caching Architecture

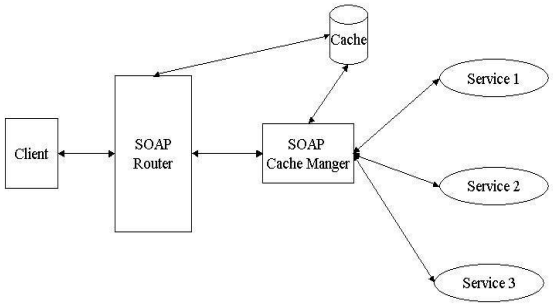

Figure 3 SOAP Caching Architecture

Cache is an abstraction of physical memory used for storing SOAP requests and responses. In this architecture (figure 3), client is responsible for invoking SOAP requests and receiving SOAP responses. SOAP Router is responsible for receiving requests from a client and redirecting these requests to the SOAP cache. The SOAP router receives SOAP responses from the SOAP cache and the SOAP router sends the responses to the client. The SOAP router is also responsible for performing administration functions like maintaining the size of the cache repository and writing cache contents from the run time memory in to a persistent storage on the hard disk when the application server on which the SOAP cache is mounted is being shutdown.

SOAP Cache Manger is responsible for storing requests and response SOAP messages in the cache. It is responsible for generating cache keys, which is used to index cached data. It is responsible for maintaining consistency of data in the cache.

4.1 Cache Keys

A cache key is used to easily locate cache data. The use of cache key simplifies the process of searching data in the cache and also helps in decreasing the time needed to search data in the cache. Two types of cache keys have been used to index cache data in the caching solution provided in this minor project. They are as follows.

Service Key: - A service key is used to reference data belonging to a particular web service. It is extracted from a SOAP request message. It is made up of the web service name.

E.g.

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <ns1:getPatientDetailsByURNO
xmlns:ns1="urn:PatientDetails">
    <name
xsi:type="xsd:string">001100006</name>
   </ns1:getPatientDetailsByURNO>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

In the above SOAP message the service key is "urn:PatientDetails" which is used to reference the web service named by Patient Details.

Message key: -A message key is used to reference data belonging to a particular method in a web service. It is extracted from a SOAP request message. It is made up of the method name and the parameter values of that method.

```
E.g.  <SOAP-ENV: Envelope>
        <SOAP-ENV:Body>
    <ns1:getPatientDetailsByURNO
xmlns:ns1="urn:PatientDetails">
    <name
xsi:type="xsd:string">001100006</name>
   </ns1:getPatientDetailsByURNO>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

In the above SOAP message the message key is "getPatientDetailsBy URNO 001100006" which is used to reference the function getPatientDetailsByURNO with parameters 001100006 under the web service named by PatientDetails.

4.2 The Hierarchy and Structure of the Cache data structures

The cache data structure is maintained in the form of a linked list. The ServiceContainer is a serializable java class used to store details of all the published web services. The ServiceContainer is made up of ServiceElements. A ServiceElement is a serializable java class used to store details of a particular web service. The ServiceElement is made up of CacheElements. Each ServiceElement is indexed by a service key. The CacheElement is a serializable java class used to store method details under the particular web service and SOAP requests and responses to that particular method. Each CacheElement is indexed by a message key (figure 4, 5). Figure 6 indicating the Hierarchy of a Cache data Structure.

4.2 Basic Workflow of the SOAP Cache

The following sequence diagram depicts the basic workflow of the SOAP cache.

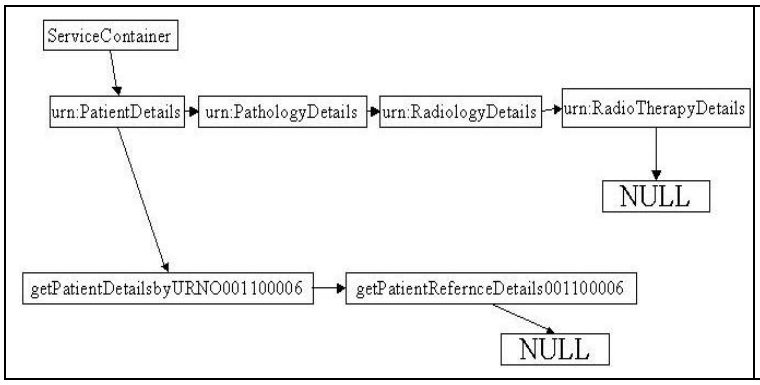

Figure 4 Example SOAP Cache Hospital Management Linked List Structure

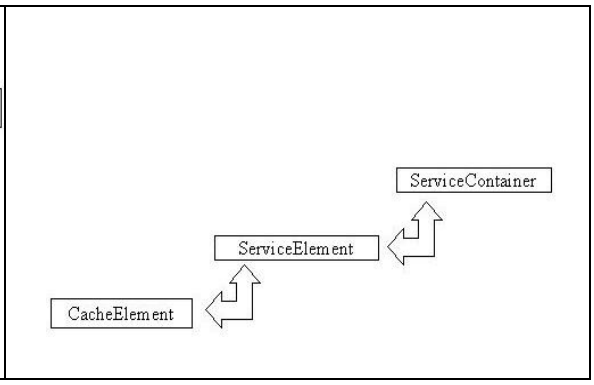

Figure 5 Sequence Diagram Showing SOAP Cache workflow

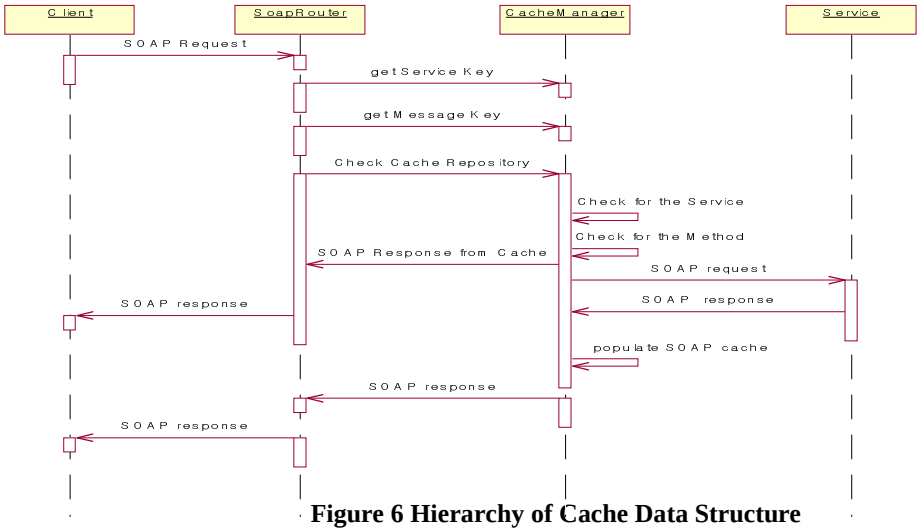

Figure 6 Hierarchy of Cache Data Structure

A client issues a SOAP request to a published service. This SOAP request is passed to a SOAP router and generates a service key, which identifies the published service and a message key, which identifies the method under the particular published service from the SOAP request message. The SOAP router invokes a function on the cache manager to check if a SOAP response is stored for the corresponding service key and message key. If present the stored SOAP response is sent to the SOAP router, which in turn is sent to the client. If not, specific entries of the service key and message key are made in the cache data structure to identify the service, the method under the service and a SOAP request is sent to the corresponding service, the SOAP response from the service is stored in the cache data structure under the corresponding service key and message key of the SOAP response and the SOAP response is sent to the SOAP router, which in turn is sent to the client.

4.3 Eviction from the SOAP cache

Eviction from the SOAP cache is carried out to decrease the size of the cache repository. Decreasing the size of cache repository results in fast retrieval of required cache objects and also prevents from storing invalid cache objects.

The Eviction policy used in the caching component of this minor project is Least Recently Used (LRU) policy and size policy. When cached SOAP objects have to be evicted, the cache repository is initially searched for cached SOAP object, which have been used less frequently used, and these cached SOAP objects are evicted. If two or more cached SOAP objects have the same least frequency of usage, then the sizes of the cached SOAP objects are compared. The cached SOAP object having the maximum size is evicted from the SOAP cache repository.

4.4 Maintaining Consistency of the SOAP cache

Figure 7 depicts, the workflow consistency of SOAP cache is maintained by using time policy. The administrator of the SOAP cache sets the expiry time for the SOAP objects to be cached. When a new SOAP object is cached, the last access time attribute of the SOAP cache object is set to the current time. When a cached SOAP object is fetched from the cache repository the difference between the current time and the last access time is calculated and compared to expiry time. If the difference is greater than the expiry time a fresh SOAP response is fetched from the web service. This SOAP response is cached. Setting the time attributes of the cached SOAP object to current time and the SOAP response is returned to the client or if the difference is less than the expiry time the cached SOAP response is returned to the client (figure 5.6.b, 5.6.c).

6. Experimental Results

The test environment consisted of three terminals. The first terminal hosted the client. The second terminal acts as an intermediate between the client and the web service. The cache module is loaded on the second terminal. The third terminal hosted the web service. The three terminals were connected in a local LAN network. The three terminals communicated with each other over HTTP. The three terminals run on a Pentium 3 500MHZ processor and a 256 MB RAM.

6.1 Testing Approach

Two types of testing methodologies were used. The first test procedure involved comparing response times for retrieving a character array of varying sizes with and without cache. The second test procedure involved comparing response times for hospital management web services workflow with and without cache. For each test data, the response time of 1000 SOAP messages were tested with and without a cache, so as to minimize the effect of external factors like network latency, IO latency etc.

The fig 8 indicates the test results for a varying size character array with and without cache. The fig 9 indicates the test results for a varying size SOAP inputs in kilobytes with and without cache. The graphs in 8 and 9 indicate that the performance of SOAP improves drastically by the use of cache. By caching SOAP requests and responses locally on the client side the overhead time involved in the construction and parsing SOAP message is avoided.

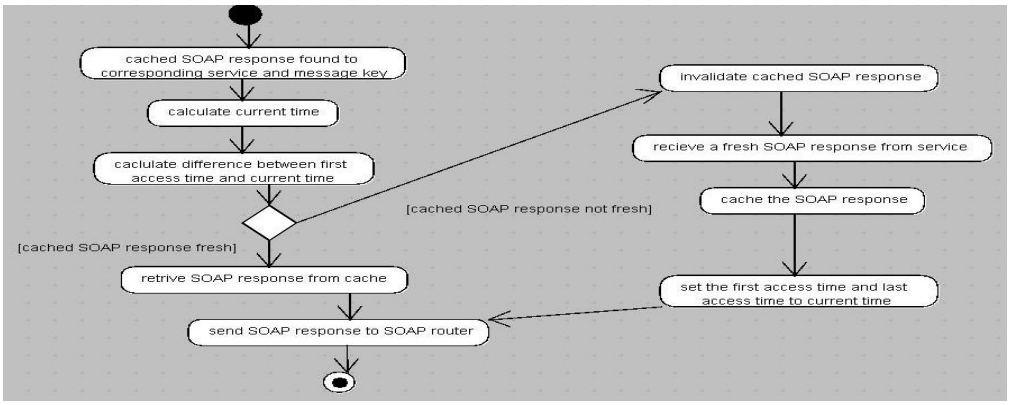

Figure 7 Consistency of SOAP Cache Activity Diagram

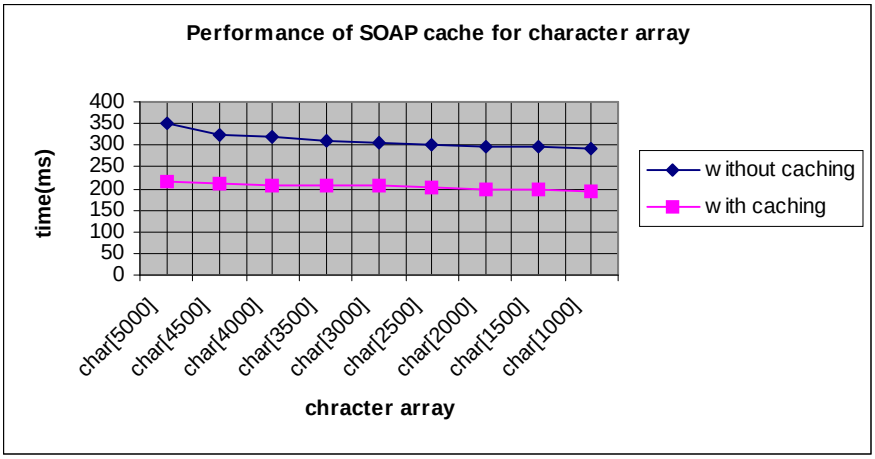

Figure 8 SOAP Performance Graph 1

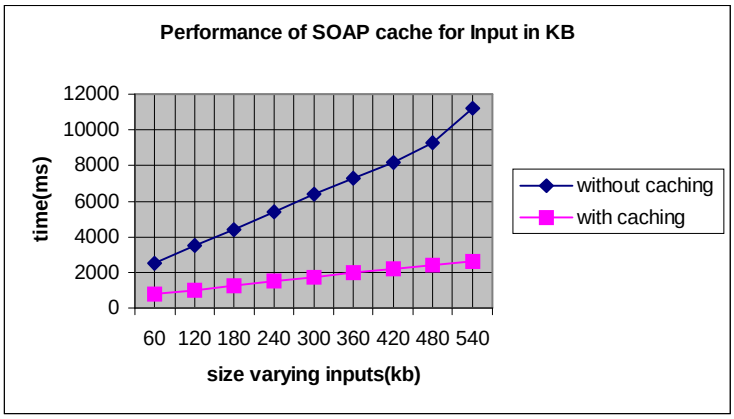

Figure 9 SOAP Performance Graph 2

6. Conclusion

This paper dealt with how caching SOAP messages can help improve the performance of SOAP. The architecture of the SOAP caching component is used in hospital management system. The data structure is used to cache SOAP objects. The eviction and consistency policies used in the SOAP cache and the testing methodology and results used to gauge the performance of SOAP by using caching.

References

[1] Meng, J. Krithivasan, R. Su, Helal, S.Y.W. S.: Flexible Inter-enterprise Workflow Management Using E-services, *Advanced Issues of E-Commerce and Web-Based Information Systems*, 2002. (WECWIS 2002). Proceedings. Fourth IEEE International Workshop on , 2002

[2] Garshol, L.: *Definitive XML Application Development*, Prentice Hall, 2002.

[3] Walsh, A. E.: *UDDI, SOAP, and WSDL: The Web Services Specification Reference Book*, Prentice Hall, 2002.

[4] Aoyama, M. Sullivan, K.: *Web services Engineering promises and challenges*, University of virgina USA, IEEE publication March 2002

[5] Chester, T. .M.: *Cross-platform Integration with XML and SOAP*, IT Professional, Volume: 3 Issue: 5, Sept.-Oct. 2001,IEEE.

[6] Box, D. Ehnebuske, D. Kakivaya, G. and et al: *Simple Object Access Protocol*, W3C Note 08 May 2000

[7] Davis, D. and Prashanth, M.: *Latency Performance of SOAP Implementation*, *Cluster Computing and the Grid* 2nd IEEE/ACM International Symposium GRID2002.

[8] Chiu, K. Govindaraju, M. and Bramley, R.: *Investigating the Limits of SOAP Performance for Scientific Computing*, Bloomington, Indiana University.

[9] Chandra, S.: *Performance Analysis of Soap*, New Jersey institute of technology, March 2002.