IAENG International Journal of Computer Science, 34:1, IJCS_34_1_19

A Smart Compressed XML Data on Networks

Xu Huang and Dharmendra Sharma
School of Information Sciences and Engineering, University of Canberra
Canberra, ACT, 2617, Australia

(Xu.Huang, Dharmendra.Sharma)@canberra.edu.au

Abstract—**It is well known that XML became an official recommendation of the World Wide Web Consortium (W3C) in 1998, it is now increasingly being used to transmit data on networks but is a verbose format and needs an efficient encoding to send relatively large amounts of data efficiently, which is most attractive to wireless data communications. It is a common technical challenge for researchers in XML-driven networks to have good performance. There are many papers discussed this issues, such as one may employ a middleware to enhance performance by minimizing the impact of transmission time [1, 3]. Normally, to reduce the amount of data sent the XML documents are converted to a binary format using a compression routine such as Gzip. However while this would reduce the amount of data, it results in an increase in the CPU time as the XML document must be compressed before being sent and uncompressed when it is received. In this paper we extended our previous research results [2, 11-15] to an enabling technology, namely Dynamic Adaptive Threshold Transmission (DATT) for XML data on networks, in particularly focusing on "dynamic" nature. We also show the experimental results obtained from our technique and that from the Network Adaptable Middleware (NAM) established by Ghandeharizadeh et al [1]. Experimental results show that our method is superior to the NAM method [1], which supported by the fact that the time taken is 220.6 times better in the applied regions.**

Keywords-XML, network adaptable middleware, efficient XML, internet middleware, efficient communication on a network.

I. Introduction

It is well known that XML has become an increasingly important data standard for use in organizations as a way to transmit data [4, 5, 6, 7] and has also attracted the attentions of those people who are working in areas of wireless communications, in particular for so called small wireless devices. Additionally it is being used to enable web services and similar, often custom, RPC functionality to allow greater access to data across multiple systems within an organization and allowing the possibility of future systems to be created from collections of such RPC functionality.

However, XML is a verbose, text based format with strict requirements on structure and is often criticized for its large space requirements. This large size can be particularly problematic for use in transmission across a network, where network bandwidth restrictions can cause significant delays in receiving the transmission, which has drawn great attention from the wireless communications.

One solution to this problem is to look at reducing the size of these transmissions by rendering them in a binary format, such as by using XMill or Gzip to compress an XML document. However such methods can take longer as compressing and decompressing may take more time than what is saved transmitting the smaller XML document.

Another solution to this problem may be the Network Adaptable Middleware (NAM) raised by Ghandeharizadeh et al [8], even though there are some ways to directly compress, such as column-wise compression and row-wise compression for large message sizes [9]. This solution estimates the time it will take to compress, transmit in binary format and decompress a document compared to an estimate of how long it would take to transmit the document as uncompressed text. The estimates are based on a persistent collection of information on how the system has performed in the past and provides an accurate estimate on whether it would be faster too compress the document before transmission or not.

We have introduced another way of determining when to compress an XML document before transmitting it in our *One Pass Technique* (OPT) and extend OPT to Dynamic Adaptive Threshold Transmission (DATT) [2, 11-15]. In this technique we determine a threshold size value for the networks. Any XML document that is smaller than this threshold will be sent uncompressed while any XML document larger than this size will be compressed before it is sent.

As we knew that the performances on networks depend on various parameters, such as traffic situations, bandwidths, transferring rates, etc. We shall use the adapted dynamic threshold to represent the characteristics of the running networks, by which the transferring XML data on networks will be controlled with the optimum condition in terms of transferring decision time defined in the next sections. The following sections are as follows, in section 2, we shall briefly review the established OPT technique and show that there is possible to improve the OPT technique. In section 3 the Dynamic Adaptive Threshold Transmission (DATT) for XML data on networks will be demonstrated, together with the experimental setup design, which is the natural research project, extended from the previous research results. We shall present the conclusion of this paper in the section 4 in particular we are going to highlight the "dynamic" issue as the extending part of our previous paper.

II. Threshold Method and its Application

Before we establish our Dynamic Adaptive Threshold Transmission (DATT) for XML data on networks, we need briefly to

(Advance online publication: 15 August 2007)

recall our previous method, titled "One Pass Technique" (OPT) and show OPT needs to be changed if we want it work well on a network. Then, we extend out previous results to current DATT in next section.

In contrast to the five network factors that contribute to the latency time of delivering a query output [1] based on the analysis of the one gigabyte TPC-H benchmark [10], our method presented here is utilizing an established " threshold" for the current working status and then to have "one-pass" transmission. We defined a threshold value for the network such that the transmitted time, for XML documents whose size has been compressed (such as via Gzip) and uncompressed, will be comparable. To determine what this value could be, we first need to determine the networks characteristics. As the networks characteristics will evolve with time the threshold value needs to dynamically change with the network.

Before OPT can be used on a network we need to determine the threshold value by making a number of XML transfers of different sizes across the network. The transmissions need to be made both with the document compressed, using Gzip as an example, (and decompressed where it is received) and by transmitting the document without compression. An estimate of how long it takes to transmit a document of a given size can then be determined by curve fitting to these results. The threshold value is set to be the size when the estimated time to transmit it without compression is equal to the estimated time to transmit it with compression. In some situations this may result in a threshold value that will require compression of all documents or one that will never require compression of a document.

There are a number of factors that can prevent OPT from yielding the best result for all cases. The threshold value will only be valid for the network bandwidth it is calculated for, so if that bandwidth changes a threshold value will give an inaccurate result and a new threshold value will need to be determined.

The compression and decompression times are dependent on the CPU load. If the load on a CPU is heavier (or lighter) than it was when calculating the threshold value it may not make the appropriate decision on whether or not to use compression on the XML document. Similarly the technique works best with a homogenous set of CPUs. Different CPUs will take different time periods to compress and decompress the XML documents. The compression/decompression time of two low end CPUs on a network will be different to the compression/decompression time of two high end CPUs on the same network using the same threshold value. This can also lead to the OPT making a wrong decision on whether or not to compress the document.

OPT can also be affected by changes in the networks traffic density. If the network is under a heavier load than it was when the threshold value was calculated the technique is more likely to transmit an uncompressed XML document when a compressed document would have been faster, and with a lighter network load compressed XML transmissions are more likely to occur when an uncompressed transmission would have been faster. OPT is best used in a homogenous environment where the network bandwidth is well known and network traffic is reasonably stable.

As we discussed that a threshold depends on many factors on the network, if OPT works for a network, it must be changed from time to time depending on the current status of the network, namely it must be dynamically changed to control the transfer date on network. This is the basic idea of our Dynamic Adaptive Threshold Transmission (DATT) for XML data on networks.

III. DATA FOR XML DATA ON NETWORKS AND EXPERIMENTAL RESULTS

It's noted that there is another solution to this problem titled Network Adaptable Middleware (NAM) [1, 8]. This solution estimates the time that it will take for the whole process, that is to compress, transmit in binary format and decompress a document, compared to the time that it would take to transmit the document without compressing it. These estimates are based on a persistent collection of information on how the system has performed in the past and provides a reasonably accurate estimate of whether it would be faster to compress the document before transmission or not.

Whenever NAM transmits an XML document it uses records of the times taken for different events to estimate how long such events are expected to take for the current XML document. NAM then decides if it should send the XML document compressed as a result of the comparison, which can be expressed as:

$$\textit{If } t_{Uncompressed\ Transmission} > t_{Document\ Compression} + t_{Compressed\ Transmission} + t_{Document\ Decompression} \quad (1)$$

Then transmit_compressed,
Else transmit_uncompressed.

where *t* is the time.

So if the time taken to transmit the XML document uncompressed is less than the time taken to compress, transmit and then decompress the document, the XML document will be sent uncompressed.

As the estimate must be calculated whenever a document is to be sent it is the case that NAM can spend a significant amount of time in determining if the document should be compressed or not. Additionally, as the estimates are based on the size of the XML document being transmitted, NAM assumes that when two different XML documents with the same physical size in bytes are compressed the result will also have the same physical size.

In order to have a Dynamic Adaptive Threshold Transmission (DATT) for XML data on networks, we make a programmed process to check the current network working situation that depends on the current traffic parameters discussed in above sections. Then, the current threshold is worked out based on the same principle described for OPT in section II. The obtained threshold will replace the previous one to work (control) the traffic communications. Since the threshold is monitored dynamically the adaptive threshold will always keep record of the times taken in transferring the data

In order to investigate our Dynamic Adaptive Threshold Transmission (DATT) for XML data on networks, we design our experimental work as shown in Figure 1.

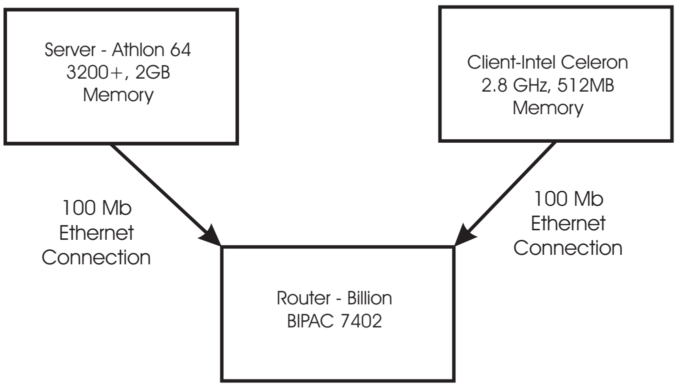
Server - Athlon 64 3200+, 2GB Memory
Client-Intel Celeron 2.8 GHz, 512MB Memory
100 Mb Ethernet Connection
100 Mb Ethernet Connection
Router - Billion BIPAC 7402

Figure 1: The experimental setup diagram for DATT

The connection was made across the router using a raw TCP connection created for each transmission. The client used a listener to listen for the incoming files (port 9013). The server was running a Cron style task scheduler to initiate communication and deliver the file.

A number of XML documents (1200 files) were gathered to test using a time based threshold as shown in Figure 2 to decide on when to compress a document and when not to. These files were of different sizes. An application program was written to transmit these documents a number of times across a network using a threshold value. Any XML document with a size greater than the threshold value is transmitted compressed while all other XML documents are sent uncompressed.

Hence we have the protocol as shown below:

If $\text{Size}_{\text{Document}} > \text{Size}_{\text{Threshold}}$ *Then* transmit_compressed, *Else* transmit_uncompressed

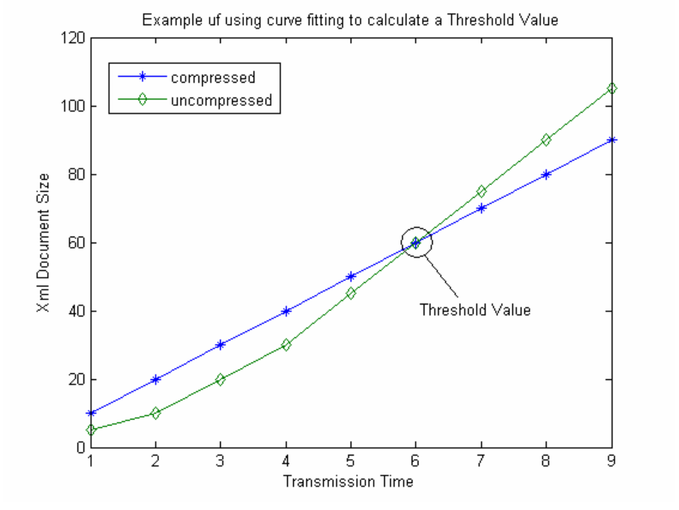

Figure 2: Curve fitting example sets of uncompressed and compressed transmission data to determine the Threshold Value.

A similar application was set up to transfer the documents using the NAM methodology (Ghandehazrizadeh, 2003). NAM uses measured network and computer characteristics to compare estimates on how long it would take to transmit an uncompressed document against an estimate of how long it would take to transmit a compressed document. The algorithm used is:

If $\text{Time}_{\text{Uncompressed Transmission}} > \text{Time}_{\text{Document Compression}} + \text{Time}_{\text{Compressed Transmission}} + \text{Time}_{\text{Document Decompression}}$ *Then* transmit_compressed, *Else* transmit_uncompressed.

We carried out the experiments with the SOAP XML documents and the CSV files. Table 1 shows the results seen when compressing an empty SOAP message.

Table 1. Size of an empty SOAP message

	Uncompressed file	WinZip compressed	WinRAR compressed
SOAP	363 bytes	311 bytes	279 bytes

Table 2 shows the compression of a SOAP message that contains a single entry. Compressing the SOAP message results in a size reduction of about 50%, however it is still significantly larger than the CSV file.

Table 2. Size of a SOAP message with one entry

	Uncompressed file	WinZip compressed	WinRAR compressed
SOAP	1,136 bytes	523 bytes	564 bytes
SOAP	773 bytes	212 bytes	285 bytes
CSV	138 bytes	221 bytes	187 bytes

Table 3 shows the compression of a SOAP message with two entries. With two data entries the size cost saving is up to 30%. The CSV file is still significantly smaller, particularly when similarly compressed.

Table 3. Size of a SOAP message with two entries

	Uncompressed file	WinZip compressed	WinRAR compressed
SOAP	1,878 bytes	622 bytes	574 bytes
SOAP, per data entry	758 bytes	156 bytes	148 bytes
CSV	298 bytes	271 bytes	240 bytes
CSV, per data entry	379 bytes	136 bytes	120 bytes

With the ten data entries seen in Table 4 the size cost saving is now around 18%. The CSV file is actually larger than the compressed SOAP file and the compressed CSV files offer only a small size cost improvement.

Table 4 . Size of a SOAP message with ten entries

	Uncompressed file	WinZip compressed	WinRAR compressed
SOAP	8,384 bytes	1,484 bytes	1,358 bytes
SOAP, per data entry	802 bytes	117 bytes	108 bytes
CSV	2,153 bytes	933 bytes	923 bytes
CSV, per data entry	215 bytes	93 bytes	92 bytes

The twenty data entries shown in Table 5 show the size cost saving is now 14% and there is decreased difference between the compressed SOAP document when compared to the compressed CSV file**.**

Table 5. Size of a SOAP message with twenty entries

	Uncompressed file	WinZip compressed	WinRAR compressed
SOAP	16,623 bytes	2,352 bytes	2,175 bytes
SOAP, per data entry	813 bytes	102 bytes	95 bytes
CSV	4,572 bytes	1,667 bytes	1,654 bytes
CSV, per data entry	229 bytes	83 bytes	83 bytes

Table 6 shows the results of a SOAP message with fifty entries. The size cost saving by compressing the SOAP message is down to 11% of the uncompressed size. The compressed SOAP document is now only about 12% larger than the compressed CSV file.

Table 6. Size of a SOAP message with fifty entries

	Uncompressed file	WinZip compressed	WinRAR compressed
SOAP	41,426 bytes	4,742 bytes	4,515 bytes
SOAP, per data entry	821 bytes	89 bytes	84 bytes
CSV	11,915 bytes	3,853 bytes	3,851 bytes
CSV, per data entry	238 bytes	77 bytes	77 bytes

Table 7. Size of a SOAP message with one hundred entries

	Uncompressed file	WinZip compressed	WinRAR compressed
SOAP	83,320 bytes	8,971 bytes	8,798 bytes
SOAP, per data entry	830 bytes	87 bytes	85 bytes
CSV	24,708 bytes	7,903 bytes	7,900 bytes
CSV, per data entry	247 bytes	79 bytes	79 bytes

The results of one hundred data entries in a SOPA message are shown in Table 7. The size cost saving is still about 11% of the uncompressed SOAP message size. The compressed SOAP document is now only about 9% larger than the compressed CSV file.

All the observations show that the efficiency of the compression applied to an XML document improves as more data instances are added. This would be expected given that the structured nature of the XML format means that each data instance would include the same opening and closing element tags. What was interesting is how the compressed SOAP XML documents began to approach the size of the compressed CSV files. As more data instances are added to the SOAP XML document and to the CSV file the difference in their compressed sizes is becoming negligible. The SOAP XML documents richly structured format would make this a preferable format for describing large sets of data.

A similar application was set up to transfer the documents using the NAM methodology (Ghandehazrizadeh, 2003). NAM uses measured network and computer characteristics to compare estimates on how long it would take to transmit an uncompressed document against an estimate of how long it would take to transmit a compressed document. The algorithm used is:

The experiment was conducted using a client PC (754pin Athlon64 3200+@2.05GHz with 1GB RAM), one Server PC (Celeron D 2.8@2.79GHz with 512MB RAM) connected by a Router (Billion BIPAC 7402G) over a 100MBit Ethernet connection.

For the DATT the time taken is calculated by the follows:

calculation time + *compression time* + *transfer time* + *decompression time* + *threshold calculation time*

In order to obtain good statistics and fair distributed results, a set of twenty-nine runs were carried out for each technique, namely DATT and NAM, sandwiched for one hour. For such a setup the whole running process covers more than 41 hours without breaking.

In order to change the working environments from time to time, the network, while it was processing, has been disturbed by various activities such as "downloading files" in different sizes, browsing the Internet, playing audio on the computer, etc. In order to determine the characteristics of the network before the applications they were run against it, solving the quadratic equations used to get the time and size estimates NAM uses in its decision algorithm and determining the threshold value for the current network traffic load for the OPT. When the threshold value was found at the particular time it will be used for controlling the XML data transferred on the network.

The results for the decision time, in terms of average for all the runs, for DATT are shown in Figure 3. As the process of determining the threshold has been passed out to a separate process, the decision times for DATT are very short.

All the running results will be recorded by five types of results, namely the original file size in bytes, decision time (for DATT and NAM, it is the time to decide whether the current file should be compressed and sent or just sent, according to the principle of DATT or NAM respectively), compression time, transferring time, and decompression time.

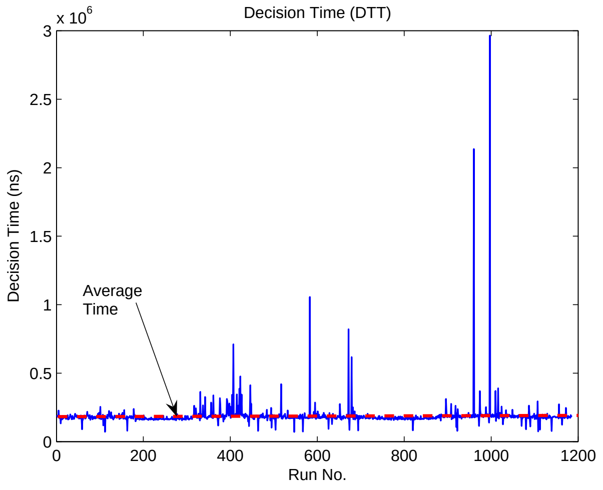

Figure 3: The experimental results for the average decision time of the DATT described in section III.

The results for the decision time, in terms of average for all the runs, for NAM are shown in Figure 4. It is seen that, as NAM accumulates more data from which to make a decision increases, the time it takes to actually make a decision increases to allow all the data to be read.

In the both diagrams, the average decision time were marked as dashed lines.

It is important to highlight two items, one is that the horizontal axis, titled as run number, for the two figures are about 1200, which are the "average" results from 41 hours running as described in above, another one is that the plotted run order should not be meanness due to the fact that we put two results, compressed files and uncompressed files together. In other words, when the input file is picked up and sent to be judged either by the DATT or the NAM, the output will be sent to two groups, one is compressed (if the file was compressed and then sent) another one is uncompressed (if the file was not compressed and directly sent). Because we do care about the average time taken by the decision rather than when the decision made for the plotting so we just put them together. Also one may find the plotting results seem to be "periodic" results, in particular for Figure 3, which, however, are not real time results.

In these experiments, the average decision time for DATT is 0.187622 milliseconds and for NAM is 41.381926 milliseconds, which means NAM takes 220.56 times longer than DATT to make a decision.

It is also a very interesting to note that the experimental results show the number of compressed files with DATT is 587 files of 1201 running files, which gives the compressing ratio = 0.488. In contrast to DATT, the NAM has 510 files compressed from total 1219 running files, which gives a compressing ratio = 0.418. Therefore, the compressing ratio NAM is about 86.4% as that of DATT. This shows that, in comparison with DATT, NAM is always (or in terms of average) making cautious decisions to keep itself in optimum states but causes heavier network traffic, which means the DATT will make higher quality network transfers for XML data on networks. This improvement for DATT against to NAM is about a quarter percent.

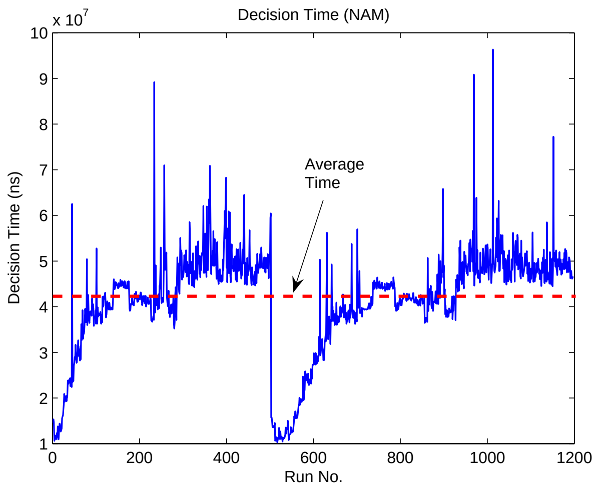

Figure 4: The experimental results for the average decision time of the NAM described in section III.

In terms of the performances for both the DATT and NAM techniques in this experimental setup they used the same software for compressing and decompressing software, but the processing is different due to the different principles used by each technique. Thus, the data shows, in terms of average, the compressing time and decompressing time are about balanced, shown by the fact that "Time taken by DATT" divided by "Time taken by NAM" is 0.91 for compressing time and 0.895 for the decompressing time. However, DATT technique is more stable processes for compressing supported by the fact that the standard deviation of compressing time is almost ten times small than that of NAM, even the standard deviation of decompressing is fairly comparable, which is understandable since for DATT when the threshold is obtained the rest of the job is much easier than that of NAM. This is further evidence to show the DATT technique will keep running networks in a better service quality.

One of the questions needs to be answered is how frequent time period is the period time taken for DATT for the nature of "dynamic" issue. Since the total time taken for DATT and NAM varies depending on the traffic situations in the networks. For example when the traffic is busy the DATT will take more time than that for NAM since the "threshold" calculation together with the comparisons then decision made will be dearer than that for NAM. In contrast the traffic situation is reasonable relaxant the time costs for DATT will be less than that for NAM due to the "threshold" calculation and comparisons will be easier. We take the traffic distributions as Poisson distributions and from the PDM of traffic distribution from 8:00 am to 8:00 pm as "effective" distribution as the common cases. The data shows that when the traffic ranges between 40% of the peck the DATT will be suggested to use otherwise the NAM will be suggested.

IV. CONCLUTIONS

We have extended our previous research results to Dynamic Adaptive Threshold Transmission (DATT) for XML data on networks. We compared this technique (DATT) to another control technique, the Network Adaptable Middleware (NAM), and found that the DATT technique is much better than NAM in terms of decision time taken, which was about 220.56 times for the DATT of the decision time less than that of the NAM's. Also the DATT will give running networks better performance by as much as one quarter in comparison with NAM. In the real life the simulation results suggest that both two methods may be combined that will give better results.

REFERENCES

[1] S. Ghandeharizadeh, C. Papadopoulos, M. Cai, and K. K. Chintalapudi, Performance of Networked XML-Driven Cooperative Applications", In Proceedings of the Second International Workshop on Cooperative Internet Computing Hong Kong, China, August 2002.

[2] Alexander Ridgewell, Xu Huang, and Dharmendra Sharma, "Evaluating the Size of the SOAP for Integration in B2B", the Ninth International Conference on Knowledge-Based Intelligent Information & Engineering Systems Melbourne, Australia, September, 2005. Part IV, pp.29.

[3] H. Liefke and D. Suciu. XMill: An efficient Compressor for XMLL Data. Technical Report MSCIS-99-26, University of Pennsylvania, 1999.

[4] Curbera, F. Duftler, M. Khalaf, R. Nagy, W. Mukhi, N and Weerawarana, S.: Unraveling the web services web: An introduction to SOAP, WSDL, UDDI. IEEE Internet Computing, 6(2): 86-93, March-April 2002.

[5] Fan, M. Stallaert, J. and Whinston, A. B.: The internet and the future of financial markets, Communications of the ACM, 43(11):83-88, November 2000.

[6] Rabhi, F.A. and Benatallah, B.: An integrated service architecture for managing capital market systems. IEEE Network, 16(1):15-19, 2002.

[7] Kohloff, Christopher and Steele, Robert: Evaluating SOAP for High Performance Business Applications: Real-Time Trading Systems, 2003, http://www2003.org/cdrom/papers/alternate/P872/p872\kohlhoff.html, accessed 22 March 2005.

[8] S. Ghandeharizadeh, C. Papadopoulos, M. Cai, R. Zhou, P. Pol NAM: A Network Adaptive Middleware to Enhance Response Time of Web Services, 2003, MASCOTS 2003: 136

[9] R.R. Iyer and D. Wilhite. " Data Compression Support in Databases." In Proceedings of the 20th International Conference on Very Large Dasta Bases, 1994

[10] M. Poess and C. Floyd. "New TPC Benchmarks for Decision Support and Web Commerece." ACM SIGMOD Record, 29(4), Dec 2000.

[11] Xu Huang, Alexander Ridgewell, and Dharmendra Sharma, "Efficacious Transmission Technique for XML Data on Networks," International Journal of Computer Science and Network Security, pp.14-19. Vol.6 No.3, March 2006.

[12] Xu Huang, Alexander Ridgewell and Dharmendra Sharma, "A Dynamic Threshold Technique for XML Data Transmission on Networks", The tenth International Conference on Knowledge-Based Intelligent Information & Engineering Systems, Bournemouth, UK, Oct 2006. B. Gabrys, R.J.Howlett and L.C. Jain (Eds) KES 2006 Part III. LNAI 4253 pp1163-1167, 2006 ©Springer-Verlag Berlin Heidelberg 2006.

[13] Xu Huang and Dharmendra Sharma, "A New Efficient Transmission for XML Data on Networks", International Multi-Conference of Engineers and Computer Scientists 2007, Hong Kong, 21-23 March, 2007. Proceedings of the International MultiConference of Engineers and Computer Sciences 2007, Volume II, pp1238-1241.

[14] Xu Huang, Alexander Ridgewell, and Dharmendra Sharma, "Efficacious Transmission Technique for XML Data on Networks," IEEE Consumer Communications and Networking Conference, Las Vegas, NV, USA. 11-13 January 2007.

[15] Xu Huang and Dharmendra Sharma, "A New Efficient Transmission for XML Data on Networks", International Multi-Conference of Engineers and Computer Scientists 2007, Hong Kong, 21-23 March, 2007. Proceedings of the International MultiConference of Engineers and Computer Sciences 2007, Volume II, pp1238-1241.