IMTC 2006 – Instrumentation and Measurement
Technology Conference
Sorrento, Italy 24-27 April 2006

Scalability Measurement of a Proxy based Personalized Multimedia Repurposing System

M. Shamim Hossain and Abdulmotaleb El Saddik
Multimedia Communications Research Laboratory
School of Information Technology and Engineering, University of Ottawa
800 King Edward, K1N 6N5, Ottawa, Canada
Fax: +1-613-562-5664, Email: {shamim, abed}@mcrlab.uottawa.ca

Abstract – Multimedia content repurposing is a new challenge in distributing multimedia systems due to the heterogeneity of client devices and their network connections. This paper addresses the problem of heterogeneity and proposes a solution that uses a series of repurposing proxies in a chain fashion. In order to find the best scalable repurposing service among those proxies, two modified versions of the shortest path selection algorithms have been implemented. The scalability of the services has been measured in terms of user satisfaction. An implemented java-based prototype as well as computer simulation results shows the viability of our approach.

Keywords – repurposing, transcoding, scalability, network characteristics

I. INTRODUCTION

The unprecedented growth of ubiquitous communication infrastructures, multimedia coding standards, and pervasive computing technologies has allowed access to personalized multimedia contents ubiquitously. This growth comes with an increasing heterogeneity of client devices as well as user preferences. The client devices range from PC to PDA, which have different device profiles in terms of resolution, power and memory. Therefore, the heterogeneity of clients and their ubiquitous connections pose new challenges in the delivery and presentation of rich personalized multimedia content to users according to their preferences at a desired scalability level. This heterogeneity problem can be solved through content repurposing.

Content repurposing refers to the conversion process by which multimedia content that is originally designed for a particular device, platform, or user, is transformed to fit other devices, platforms, or users [1]. Sometimes, it is interchangeably referred as content transcoding or adaptation, which means conversion of one format into another format. This conversion or repurposing can be related to a coding standard (MPEG to H.263), a coding parameter (e.g. frame rate, bit or spatial resolution), and/or personalization (e.g. based on the user or natural environment profile).

Depending on the location where the repurposing takes place, content repurposing can be classified into three categories: server-based, client-based, and proxy-based. In the server-based approach [11], the content server is responsible for performing the trans-coding; the content provider has all the control on how the content is trans-coded and presented to the user. A server-based adaptation does not scale properly for a large number of users and it requires a high-end content and delivery server to handle all requests. As for the client-based approach [12], the client does the trans-coding on the fly (immediately upon receipt). The advantage of this approach is that the content can be adapted to match the characteristics of the client exactly. But, at the same time, client-based adaptation can be highly expensive in terms of bandwidth and computation power, especially for small devices with small computational power and slow network connectivity, with large volumes of data that might be wastefully delivered to the device to be dropped during repurposing. In the proxy-based approach [1, 13-16, 18]; the proxy server is located on the network between the server and the client. The proxy, which makes requests to the server on behalf of the client, receives content responses from the server, analyzes and repurposes the requested content on the fly, and finally sends the repurposed content back to the client, based on its capabilities. Proxy adaptation has a number of benefits [4], including improved opportunities for proxy caching [2, 3], improved client-perceived latency, leveraging the installed infrastructure, and scaling properly with the number of clients. It also provides a clear separation between content creation and content adaptation. Therefore, the proxy based approach is a better solution for the successful distribution of multimedia content based on the user and device profile.

Due to the nature of the real-time rich multimedia content and its high complexity of repurposing, the proxy-based repurposing approach [2, 10] is the most suitable one. To reduce this complexity, a series of repurposing proxies can be chained together [17, 20]. Now the issue is to find the best sequence of repurposing paths, between the sender and the receiver that can maximize the scalability and is renderable to heterogeneous clients. This can be done through using a proper path selection algorithm. The algorithms proposed in this paper use the scalability factor (user's satisfaction) in terms of repurposed content quality as the optimization metric for the path selection function. The proposed work is different from that of the work in [20], in selecting and creating a sequence of repurposing paths to match the capabilities of the sender and receiver. The performance analysis we conducted shows that our approach is more scalable [8] in terms of the user's satisfaction and in selecting repurposing paths based on the end result. Before developing

a real prototype, a java based computer simulation is implemented and tested, using two algorithms, in order to check the scalability. The performance result shows the viability of the system.

II. PROPOSED RESEARCH

The primary function behind content repurposing is to enable a diversity of client devices, with different restricted capabilities, to access personalized multimedia content. Personalization refers to the repurposing based on the user's preferences, and locations. Device constraints and heterogeneity are not the only limiting factors for the successful distribution of multimedia content; personalization, and scalability (i.e. user's satisfaction) are also essential for the successful distribution of multimedia content. Therefore, during the design and implementation of a personalized multimedia repurposing system, attention should be paid to the issue of personalization in terms of the user's satisfaction.

A. System Architecture

The proposed system architecture shown in Fig. 1 comprises the following three key components:
Server: The multimedia content server holds or collects all the profiles such as the user's preferences, services provided by the server, the client's capability, proxy information and network characteristics. Upon connection with the proxy, all profiles are created in the XML format. [Appendix B: example of a profile]
Proxy: Proxies communicate with both the server and the clients. Repurposing proxies may contain a number of repurposing services. The scalable, shortest path algorithms find the appropriate path among the proxies and repurposing services.
Clients: The heterogeneous client ranges from PC to PDA, which receives the final repurposed multimedia content.

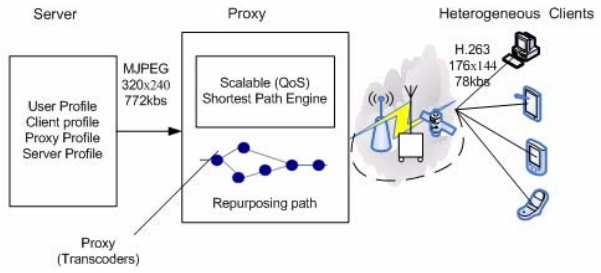

Fig. 1. System architecture.

B. System Functionality

The system functions are described in the following sequence diagram as shown Fig. 2. When the repurposing is requested each proxy can create a number of repurposing services on the fly. Once streaming through a repurposing service stops, the reception port of the proxy is freed and the repurposing object is destroyed. A stream can pass through multiple proxies and repurposing services, and if necessary, it can pass through the same proxy twice, using two different repurposing services. However, it cannot use the same proxy twice in a row in the repurposing chain. The initial connection between the client and the server is created using the SIP protocol, which is described in Fig. 3. The client sends its device information. Once the server has the client's information, it requests the information from all the proxies it knows. The server continues requesting information from the proxies for as long as it still finds proxies. Once all the information has been gathered, the server generates the graph, and finds the best scalable path based on the scalability factor using SDP (Service Discovery Protocol). The server then sends the path information and the stream to the first proxy in the chain. The proxy, upon reception of both path and stream, creates the repurposing service and sends the rest of the path (with its own information extracted) to the next proxy in the chain. The repurposed stream is also sent to the next proxy. The last proxy in the chain simply forwards the results of its repurposed contents to the client.

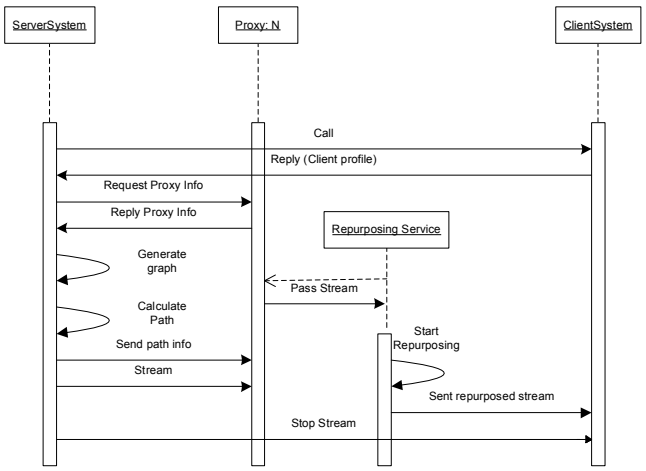

Fig. 2. Sequence diagram for the overall repurposing system.

Initially the server sends a SIP invite message to the client containing the available ports and the IP of the server. The client replies with a 200 OK message to which it adds its device information – capabilities and the IP of the devices. Once the client information is received, the server writes it to an XML [Appendix -1] file in order to store the client's profile. The server then connects to each of its neighboring proxies, and from each it requests the main information file of the proxy (which contains the proxies' neighbors), the repurposing directory of the proxy and each of the repurposing services specified in the directory. Once the information from all the proxies is received, the server calculates the best path, and starts streaming the captured video through the chosen proxies and repurposing services using the Real Time Protocol (RTP) [26].

C. Scalability Measure and Evaluation

After the creation of the profiles, scalability criterion is computed using an equation (3) and a scaling path is constructed. The path selection algorithm finds the appropriate optimal scaling repurposing path between the sender and the receiver based on the scalability metric.

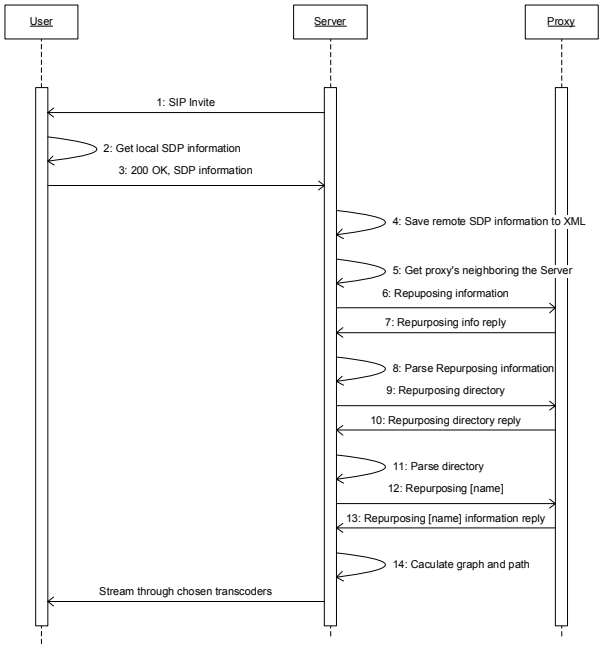

Fig. 3. Sequence diagram for the SIP communication.

The description of *repurposing services* is mentioned in the repurposing profile [Appendix B]. This profile contains a copy of the list of inputs (e.g. different formats, standards) that the repurposing service can accept and a list of outputs (e.g. formats, standards etc.) that the repurposing service can create by repurposing one of the inputs. Again, for simplification it is assumed that any of the inputs can be repurposed into any of the outputs. Both inputs and outputs are described using the MPEG-7[25] content descriptors. For example, the repurposing service can send MPEG-4 with a resolution of 352×240, and a frame rate of 30 fps as input, and it can receive H.263 with a resolution of 88×72, and a frame rate of 10 fps as output.

Scalability is the ability of a system to operate in a satisfactory manner in different scales where satisfactory implies a desirable quality of service [19] in terms of throughput and frame rate [9]. In our implementation, scalability metric k = 1> F_{max}=30 fps (frames per second) has the range of 0 to 1. Thus, the scalability function is a linear function that grows between 0 and 1 for the domain [0.30], and then becomes a constant function with the value of 1 for the values higher than 30. The frame rate itself is also calculated as a function that depends on the multimedia compression standard and the available bandwidth between neighboring computers. In order to be able to calculate the frame rate, we have selected a number of properties that could be used for each type of standard. In order to simplify the work even more, a table is chosen for the video standard and another table for the audio standard that will specify the bandwidth required for different standards with different parameters. By observing how the bandwidth changes for different parameters, we then interpolated the bandwidth required for the given values.

In this paper, scalability is determined based on the user's preferences, which are the perceived frame rates. This scalability (k) metric can be computed by using the following equations:

$$C = \sqrt{A/S}, \quad B_r = C.B \tag{1}$$

$$F_a = (B_s / B_r) \times F_{\max} \tag{2}$$

$$k = \frac{1}{F_{\max}} \times F_a \tag{3}$$

Where, A and S are the resolution of the video stream and the stored video standard (e.g. H263, MPEG-4) respectively. Also, B is the stored video's bandwidth, B_s is the available bandwidth for the video stream, and F_{max} is the available frame rate for the video stream.

D. Repurposing Graph Creation, Simplification and Path Finding

After computing the scalability metric, an acyclic repurposing graph is created and simplified by ensuring that no infinite loop exists. After simplification, the next step is to find the series of repurposing services that provide scalability (user satisfaction) to the receiver. This is done by successive comparisons of edge costs, which, for this application, is the scalability metric. As the scalability depends on the content, the repurposing selection algorithm should always compute the scalability in terms of user satisfaction after selecting each repurposing service towards the scaling path. Algorithms, which are used to find the best scaling path, are the adapted versions of Dijkstra's shortest path algorithm and Bellman ford's algorithm. RPS_1 and RPS_2 as described in [Appendix A], are the modified versions of Dijkstra [7] and Bellman ford [5] respectively. The algorithms are modified in order to incorporate them into the multimedia repurposing system. In the algorithm, scalability (weight) has been maximized instead of minimizing. After execution of the algorithms, both algorithms generate the best scaling path from sender to receiver and, at last, scalability is computed.

III. RESULTS AND DISCUSSION

The main goal of the scalability performance comparison is to show that the scaling selection algorithm demonstrates efficient means for personalized repurposing. After the implementation of the algorithms, computer simulations are

conducted. The tests are conducted 20 times by using 15 proxies and 50 different repurposing services. The time is measured in order to generate a repurposing graph and to find the path for a service. The tests are conducted for different proxies' ranging from 1 to 15. The followings are some measured findings:

- Only the number of repurposing services determines the time required to create a repurposing graph and to calculate the path. For a test with 20 repurposing services running on 8 and 9 proxies, the average graph creation time is 1525 milliseconds and 1575 milliseconds respectively. Therefore, increasing the number of proxies does not influence a lot on the time needed to create a repurposing graph.
- The RPS_1 is more scalable than that of RPS_2 shown in table 1, as users are more satisfied with the result of algorithm two.

Table I. Performance comparison using 15 proxies and 50 repurposing services.

	RPS_1	RPS_2
Maximum Scalability	0.66	0.53
Average Graph Creation time (msec)	2032.41	1993.75
Repurposing Path Finding time (msec)	10	11

- The path finding time in both of the scaling algorithms is almost 10, however in [20] it was approximately 500ms.
- With the increase of repurposing services (nodes), the graph creation time also increases which shows a linear relationship [2] as shown in Fig. 4.

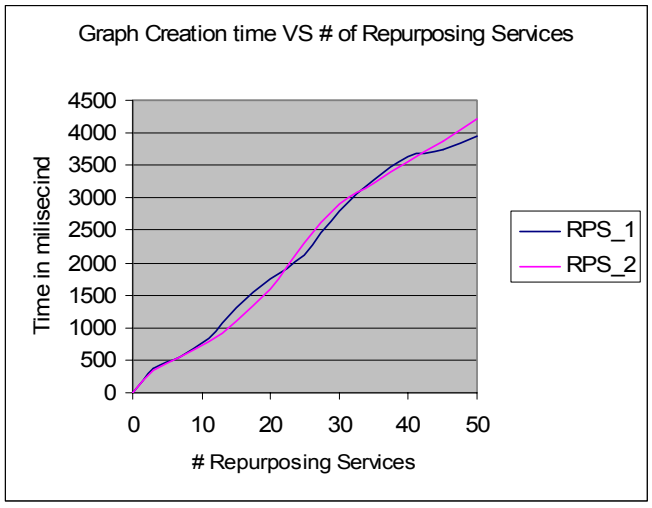

Fig. 4. Linear Relationship between Repurposing Services and Graph Creation time

After testing the algorithms' performances, a real prototype for a multimedia repurposing application system has been implemented. This prototype is built using Java because of the code portability. In this prototype, the repurposing selection algorithm is incorporated into the server in order to generate a repurposing graph and find the appropriate repurposing service from sender to receiver. The list of services is created as a long string that holds, for each service, the type (audio or video), the format, the properties of the format and the IP address, and the port number of the service. Since the scalability selection algorithm is used on the server, the client no longer needs to determine which service is the best; the client only needs to send all available services to the server. The details of creating a scalable proxy path has been discussed in Section II.B

The implementation is done using Java Multimedia Framework (JMF) [21] for the repurposing and streaming of the multimedia content, Session Initiation Protocol (SIP) [22] as the signaling protocol between the two different clients and Service Discovery Protocol (SDP) [23] for the service discovery with a wireless connection. The testing application comprises of one server (PC) for sending, two (PC) proxies and one client PC.

The application shows very good results by taking a video stream of 1000kbs and repurposing it to 30 kbs, by changing the format from M-JPEG to H.263, this causes the final resolution to be reduced by half of its original one. The video quality is very good. The real time implementation for the PDA was performed but the quality was a little bit worse than the PC.

It is found that our approach is valid and it would not take too much time to determine the repurposing path and to stream the content through a large number of proxies. Although the path-finding algorithm requires some time for a large number of proxies, this approach shows that content repurposing is a valid solution to the problem of heterogeneity. The repurposing time was very small for each proxy and the final content required a much smaller bit rate than the original one. Sometimes the reduction in the bit rate for the repurposed content was more than ten times that of the original content.

ACKNOWLEDGMENT

Authors wish to thank the NSERC funded LORNET project for its financial support.

REFERENCES

[1] A. El Saddik and M. S. Hossain, "Multimedia Content Repurposing," in Encyclopedia of Multimedia, B. Furht, Ed. Springer Book Series, ISBN: 0-387-24395-X, 2006.

[2] Ardon, P. Gunninberg, B. Landefelft, Y. Ismailov, M. Portmann, S. Sereviatne, " MARCH: A distributed content adaptation architecture", Intl. J. Commun.. Syst,, vol. 16, pp.97-115. 2003

[3] A. Maheshwari, A. Sharma, K. Ramamritham, P. Shenoy, "TransSquid: Transcoding and caching proxy for heterogeneous e-commerce environments," in Proc. 12th IEEE Int. Workshop Research Issues in Data Engg pp.50-59, Feb. 2002.

[4] B. Knutsson, H. Lu, J. Mogul, B. Hopkins, "Architecture and Performance of Server-Directed Transcoding," ACM Trans. Internet Tech., vol. 3, no. 4, Nov. 2003.
[5] Bellman-Ford algorithm. [Online]. Available: http://en.wikipedia.org/wiki/Bellman-Ford_algorithm
[6] C. Canali, V. Cardellini, R. Lancellotti, ``Squid-based proxy server for content adaptation," Dept. of Computer Engineering, University of Roma, Tor Vergata, Tech. Rep. TR-2003-03, Jan. 2003.
[7] Dijkstra'sAlgorithm [online]. Available: http://ciips.ee.uwa.edu.au/~morris/Year2/PLDS210/dijkstra.html,
[8] J. R. Smith, R. Mohan, C.-S. Li, "Scalable Multimedia Delivery for Pervasive Computing," in Proc. ACM Multimedia '99, Oct. 30 - Nov. 5, 1999
[9] J P. Jogalekar and M. Woodside, Evaluating the Scalability of Distributed Systems, IEEE Trans. Parallel and Dist. Syst., vol. 11, no. 6, June 2000.
[10] J. R. Han, P. Bhagwat, R. LaMaire, T. Mummert, V. Perret, J. Rubas, "Dynamic adaptation in an image transcoding proxy for mobile WWW browsing ", IEEE Personal Communications, vol. 5, no. 6, Dec. 1998.
[11] R. Mohan, J.R. Smith and C.S. Li, "Adapting Multimedia Internet Content for Universal Access," IEEE Trans. on Multimedia, vol. 1, no. 1, pp. 104–114, 1999.
[12] S. Björk, L.E. Holmquist, J. Redström, I. Bretan, R. Danielsson, J. Karlgren, and K. Franzén, "WEST: a Web browser for small terminals," in Proc. 12th annual ACM symp. User inters. Soft. and techn., p.187-196, November 07-10, 1999.
[13] S. Bandaru and M. Mandal, " Content Adaptation Architecture with Efficient Usage of Cached Data in a Multimedia Proxy Server," in Proc SPIE, Vol. 5242, Internet Multimedia Management Systems IV, John R. Smith ed., Sethuraman Panchanathan, Tong Zhang (SPIE, Bellingham, WA, 2003)
[14] S. Chandra and C.S. Ellis, "JPEG Compression Metric as a Quality Aware Image Transcoding," presented at the 2nd Symp. Internet Techn. and Syst. (USITS '99), Oct 12, 1999
[15] S. Chandra, C. Ellis, and A. Vahdat, "Application-Level Differentiated Multimedia Web Services Using Quality Aware Transcoding," IEEE J. Selected Areas in Commun, 2000.
[16] V. Cardellini, P.S. Yu, Y.-W. Huang, "Collaborative Proxy Systems for Distributed Web Content Transcoding," presented at the 9th Int'l ACM Conf. on Inform. and Knowledge Management, Washington, DC, Nov. 2000.
[17] W. Y. Lum and F. C. M. Lau "On Balancing between Transcoding Overhead and Spatial Consumption in Content Adaptation," in Proc. MOBICOM'02, pp.239-250, September 23–26, 2002.
[18] W. Y. Lum and F. C. M. Lau "A context-Aware decision engine for Content Adaptation," IEEE Pervasive Computing, vol. 1, no. 3, pp. 41-45, July-Sept. 2002
[19] W.Y.Lum and F.C.M Lau, "A QoS-sensitive content adaptation system for mobile computing," in Proc. COMPSAC- 2002, pp.680-685, Aug. 2002.
[20] Z. M. Mao, H. Wilson So, B. Kang, and R. H. Katz, "Network Support for Mobile Multimedia using a Self-adaptive Distributed Proxy," presented at the 11th Int. Workshop Network and Operating Syst Support for Digital Audio and Video (NOSSDAV-2001).
[21] Java Media Framework API, [online] Available: http://java.sun.com/products/java-media/jmf/index.jsp, last visited Feb. 2006
[22] Session Initiation Protocol (sip) Charter, [online] Available:http://www.ietf.org/html.charters/sip-charter.html, last visited Feb. 2006
[23] Simple Service Discovery Protocol Draft, [online] Available: http://www.upnp.org/download/draft_cai_ssdp_v1_03.txt,
[24] The Official Bluetooth Wireless Information Site, [online] Available: http://www.bluetooth.com/
[25] MPEG-7 Overview [online] http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm
[26] RTP: A Transport Protocol for Real-Time Applications, ftp://ftp.rfc-editor.org/in-notes/rfc3550.txt

APPENDIX –A

Repurposing Path Selection Algorithm (RPS_1):

1. Let VS = {Sender} be the set of all visited repurposing service. Let UV be the set of all downstream neighbors of Sender.
2. If UV is empty, then TERMINATE (FAILURE).
3. Compute the perceived Scalability (frame rate) for the repurposing services in UV
4. Select the Repurposing service Ri that has the highest scalability, add it to VS and remove it from UV.
5. If the selected Ri is the Receiver node, then GOTO Step 8.
6. Add to VS all the Repurposing Service to which Ri is directly connected.
7. GOTO Step 2.
8. Print Repurposing path from the Sender to Ri.

Repurposing Path Selection Algorithm (RPS_2):

1. If graph size is 1 TERMINATE (FAILURE).
2. Set Sender scalability=0; all other Scalability = -∞
3. Loop n times where n = number of Repurposing Services (RP) and for each iteration relax the graph:
   a) Calculate for each Repurposing Service newweight (RP) = max (weight (RP y) +weight (edge (RP y, RP))) for all Repurposing Services (RP y) that have link between themselves and repurposing service.
   b) If newweight(RP) > previous weight (RP)
      I. weight(RP) = newweight(RP)
      II. add RP y to path node
4. Try and repurposing relax graph one more
   a. If graph is relaxable TERMINATE (NEGATIVE LOOP)
5. Print Repurposing path from Sender to Receiver

APPENDIX-B

Repurposing Service profile

```
<?xml version="1.0" encoding="UTF-8"?>
<rep_service xmlns:xsi='http://www.w3.org/2001/XMLSchema-
instance'
xsi:noNamespaceSchemaLocation='file:///j:/ rep_serviceV1.xsd'>
<input>
<inputName>video/jpeg</inputName>
<iproperty>
<property>resolution</property>
<value>160x120</value>
</iproperty>
</input>
<input>
<inputName>video/h263</inputName>
<iproperty>
<property>resolution</property>
<value>352x288</value>
</iproperty>
</input>
<output>
<outputName>video/h263</outputName>
<oproperty>
<property>resolution</property>
<value>176x144</value>
</oproperty>
<visualQuality>
<videoFormat>H.263</videoFormat>
<Frame height="144" width="176"/>
</visualQuality>
</output>
<repurposeHardware>
<processor>
<processorName>K7</processorName>
<cyclesPer>700000</cyclesPer>
</processor>
<memory>256</memory>
</ repurposeHardware >
<proxyNode location='file:///j:/AONIT09.xml' />
</ rep_service >
```