Efficient Shortest Path Routing Algorithms for Distributed XML Processing

Ye Longjian[1], Hiroshi Koide[2], Dirceu Cavendish[3] and Kouichi Sakurai[4]
[1]*Graduate School of Information Science and Electrical Engineering, Kyushu University, 744 Motooka, Nishi-ku, Fukuoka, Japan*
[2]*Research Instiute for Information Technology, Kyushu University, 744 Motooka, Nishi-ku, Fukuoka, Japan*
[3]*Network Design Research Centre, Kyushu Institute of Technology, 680-4 Kawazu, Iizuka, Fukuoka, Japan*
[4]*Faculty of Information Science and Electrical Engineering, Kyushu University, 744 Motooka, Nishi-ku, Fukuoka, Japan*

Keywords: Distributed XML Processing, Web Service Processing, Web Service Processing Offload.

Abstract: This paper analyses the problem of efficiently routing XML documents on a network whose nodes are capable of distributed XML processing. The goal of our study is to find network paths for which XML documents' transmission will result in high likelihood that a large portion of the documents be processed within the network, decreasing the amount of XML processing at documents arrival at the destination site. We propose several routing algorithms for single route and multipath routing and evaluate them on a distributed XML network simulation environment. We show the benefits of the proposed XML routing algorithms as compared with widespread minimum hop routing strategy of the Internet.

1 INTRODUCTION

Web services has become an indispensable infrastructure for our society. A multitude of Web services are currently deployed, demanding significant resources, such as CPU power and memory space, to support their services quality. The processing of Web services are generally provided at end points only in the current Web services. With widespread usage of cloud processing, we envision a distributed Web service processing approach, where part of Web server's processing can be offloaded to intermediate nodes in network, reducing end points load and improving services throughput. This Web service network processing paradigm may lead to efficient resource usage with effective scheduling and higher quality of services. Currently, XML data is one of the basic communication formats in Web services infrastructure. Servers typically process XML data in various Web services (e.g. collaborative services). (Cavendish and Candan, 2008) has proposed a distributed XML processing platform, using intermediate network nodes besides clients and servers. They have shown from an algorithmic point of view how to turn well-formedness, grammar validation, and filtering into distributed processing tasks. In a subsequent work, (Y. Uratani and Oie, 2012) has studied distributed XML processing performance of network nodes capable of XML document processing, for various network topologies and document types.

This paper addressed the problem of routing XML documents across a network where nodes have various XML processing capabilities. Our objective is to find routes at which a large amount of XML processing be executed by network nodes, saving processing efforts at destination nodes. With the objective of increasing XML processing at network nodes, we tackle the problem of how best route XML traffic, and introduce single path available capacity (ACAP) and latency available capacity routing (LACAP), as well as multipath N-route shortest path and N-route disjointed shortest path XML routing algorithms.

The paper is organized as follows. In section 2, we address related work. In section 3, we describe a distributed XML processing network environment, where nodes are able to execute basic XML processing tasks, such as well-formedness checking and grammar validation. We also define the XML routing problem, and introduce various routing algorithms. In section 4, we describe a distributed XML processing experimental environment, and characterize XML routing performance. In section 5, we summarize our findings and address future research directions.

Longjian, Y., Koide, H., Cavendish, D. and Sakurai, K.
Efficient Shortest Path Routing Algorithms for Distributed XML Processing.
DOI: 10.5220/0008162002650272
In *Proceedings of the 15th International Conference on Web Information Systems and Technologies (WEBIST 2019)*, pages 265-272
ISBN: 978-989-758-386-5

2 RELATED WORK

We assume a distributed XML processing paradigm, where intermediate nodes are capable of XML processing. Recent support for this paradigm is as follows. Active Network (Tennenhouse and Wetherall, 2007) also focuses on processing at intermediate switches. The processing manner is assumed to be of two types: type-1) encapsulated into a packet, type-2) assigned to network switches beforehand. The type-1 manner executes only simple processing but is able to execute fast processing because of hardware execution. The type-2 manner, which is more similar to our system, is able to process more complex XML tasks. VNode (Y. Kanada and Nakao, 2012) also provides a processing environment at intermediate switches. In their research, the processing function is also provided at customized switches and its processing environment is provided as a virtualized environment using a virtual machine. These researches provide not only transport functions but also processing functions in the network.

Next we show current researches of specific processing in networks. In transcoding (S. H. Kim and Ro, 2012), a content server delivers data (e.g. video data) to clients via a transcoding server. The transcoding server transforms the original data to data which reflects user's demands. For instance, the data may be transformed from high resolution to low resolution at the transcoding server to adapt to mobile devices. Another intermediate node processing is a cache server (S. Nishimura and Ikenaga, 2012; Kalarani and Uma, 2013), a key technology of content delivery networks. Cache servers are allocated to wide distributed places and store contents as cache from other content servers, driven by user request patterns. Upon user's request, data is delivered from the nearest cache servers leading to lower network latency. (Fan and Chen, 2012; Solis and Obraczka, 2006) focus on sensor networks. These researches propose to consolidate the large amount of sensing data at some intermediate nodes before large data reach data collection servers. Such approach can reduce energy consumption and network load for mobile sensor devices. (M. Shimamura and Tsuru, 2010) studies the compression of packets near a sender, expanding the packets near a receiver during buffer queueing time to achieve better network resource utilization. In these works, we see that the network provides special functions such as video transformation, data caching and so on for specific services.

Finally, few papers have addressed network routing problem for efficient XML processing. (Ziyaeva and Min, 2008) addresses the problem of routing XML content to appropriate recipients within an Enterprise Service Bus (ESB), where specific XML processing is required. However, in contrast to our work, XML processing is executed only at the recipient's site. In (Wang and Ozsu, 2007), the authors address the problem of routing XML queries in an efficient way over a large Peer-to-Peer network. The objective there is to best satisfy XML queries at destination nodes, as opposed to XML execution at intermediate nodes.

3 DISTRIBUTED XML PROCESSING

Distributed XML processing requires some basic functions to be supported:

- **Document Partition:** The XML document is divided into fragments, to be processed at processing nodes.
- **Document Annotation:** Each document fragment is annotated with current processing status upon leaving a processing node.
- **Document Merging:** Document fragments are merged so as to preserve the original document structure.

XML processing nodes support some of these tasks, according to their role in the distributed XML system. XML document processing involves stack data structures for tag processing. When a node reads a start tag, it pushes the tag name into a stack. When a node reads an end tag, it pops a top element from the stack, and compares the end tag name with the popped tag name. If both tag names are the same, the tags match. The XML document is well-formed when all tags match. In addition, in validation checking, each node executing grammar validation reads DTD files, and generates grammar rules for validation checking. Each node processes validation and well-formedness at the same time, comparing the popped/pushed tags against grammar rules. Details of these node distributed processing is described in (Cavendish and Candan, 2008; Y. Uratani and Oie, 2012).

3.1 XML Routing

An overlay XML network presents networking nodes with various XML processing capacities. One way to capture node processing capabilities is to define a node processing capacity of X XML tags per second ($C = X/sec$). That being the case, a routing problem may be defined as follows. For each XML document to be transported across the network between source S and destination D, we wish to find the route such that the XML document gets the most XML processing at the network, saving destination D processing resources. Obviously, a favorable route is the one which contains many XML processing nodes with high processing capacity.

The majority of routing algorithms in the Internet use a so called minimum-hop scheme, in which a path of minimum number of network hops is computed. The rationale is that a min-hop path consumer less network resources to transport the data. Being "blind" to XML processing capabilities of network nodes, this routing scheme may be detrimental to our goal of maximizing XML processing in the network.

3.2 Minimum Hop Routing and XML

Figure 1 illustrates how a minimum hop routing may prevent XML processing offloading. In the network, only node 3 has XML processing capabilities. A minimum hop routing algorithm may route XML documents from node 0 to node 4 via route 0-1-4 only, with no network node capable of XML processing, whereas route 0-2-3-4, with three hop count, could be used for XML processing at node 3.

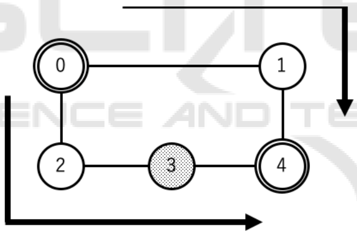


Figure 1: Example of Network for Minimum Hop Routing.

3.3 Dijkstra XML Routing Algorithm

The first XML routing scheme we propose is as follows. Lets define a link weight cost as:

$$w(v_i, v_j) = \frac{MaxXMLCap}{1 + C_{v_j}} \tag{1}$$

where $MaxXMLCap$ is the highest XML processing capacity among all network nodes, while C_{v_j} is the processing speed of each intermediate node v_j. With this definition, if node v_2 has XML processing capacity of 0, all network links arriving at v_2 will have link cost of MaxXMLCap. For nodes with higher XML capacities, their incoming link costs are less than $MaxXMLCap$. A Dijkstra routing algorithm (Cavendish and Gerla, 1998) can then be used to find the path of minimum cost, where the cost PC_p of a path p is defined as the sum of the costs of its links, or:

$$PC_p = \Sigma w(v_i, v_j) \iff w(v_i, v_j) \in p \tag{2}$$

3.4 Single Path Routing Algorithm

3.4.1 Capacity Routing

Link cost definition 1 is not practical because it requires the computation of the maximum XML capacity among all XML processing capable nodes. A more convenient way to define link cost weight is as follows:

$$w(v_i, v_j) = \frac{1}{1 + C_{v_j}} \tag{3}$$

In this equation, $w(v_i, v_j)$ is the weight of node v_j. For the nodes without any capability of processing, C_{v_j} is 0, resulting on a weight of 1, which is larger than the weights of XML processing capable nodes. We call a Dijkstra routing algorithm using this weight Capacity Routing.

3.4.2 Available Capacity Routing

Equation 3 does not take into account current XML processing load of a node, which could lead to processing overload. To take into account processing load, we adjust weight computation as follows. new calculation method called Available Capacity Routing whose weights can change over time. This calculation method is defined as follows:

$$w(v_i, v_j) = \frac{1}{1 + C_{v_j} * An} \tag{4}$$

An is an available capacity parameter, $0 \leq An \leq 1$. As such, a heavy loaded node will result in higher weight than a lightly loaded node, causing the routing algorithm to select routes around the heavy loaded node..

3.4.3 Latency Available Capacity Routing

Network latency may affect the quality of the transport service. In this case, an efficient routing strategy seeks to minimize the total network latency, rather than minimum hop count. If we define $l(v_i, v_j)$ as the link latency between nodes v_i and v_j, then the total latency of a path PC_p becomes:

$$PL_p = \Sigma l(v_i, v_j) \iff l(v_i, v_j) \in p \tag{5}$$

However, in our XML routing problem, there is a tradeoff between routes with low path latency and high XML network processing. This is because low latency paths correlate highly with low number of XML processing capable nodes. In other words, a low latency routing strategy may result in poor network processing performance. That being the case, a routing strategy that favors high latency paths may spread the XML processing around more XML processing capable nodes, improving performance. Notice that this strategy will impact end-to-end network latency, but we argue that Web Service applications tolerate a reasonable amount of extra latency, differently from real-time applications such as networked interactive gaming. Therefore, we adjust the link weight computation as follows.

$$w(v_i,v_j) = \frac{1}{1+C_{v_j} * An * l(v_i,v_j)} \tag{6}$$

Notice that as $l(v_i,v_j)$ gets larger, the weight decreases. A limitation of this weight definition is that if two nodes have two parallel links, the path including the longest link is selected during path computation. However, in this corner case node v_i may independently send traffic via the shorter link to v_j, if link bandwidth is available. In high speed networks of today, link bandwidth may be considered infinite for XML processing, as the amount of XML data is insignificant as compared to link capacities of tens of gigabits/sec. This is also the reason that we do not use bandwidth as part of a routing strategy in our work.

3.5 Multiple Path Routing Algorithm

We have introduced three weight calculation methods:CAP (Capacity Routing), ACAP (AvailableCapacity Routing) and LACAP (Latency Available Capacity Routing) which are used in single routing algorithm. To make best use of XML processing capacity of intermediate nodes, we propose multiple path routing as follows. We have two types of multiple path routing algorithms: N-route Shortest Path Routing which uses Dijkstra's algorithm to compute not only the shortest path but also second, third shortest, to as many paths as desired; N-route disjointed shortest path routing which is the same as previous multipath routing except that it avoids repeated usage of a node in different paths. These two multiple path algorithms will be introduced detailed in 3.5.1 and 3.5.2.

3.5.1 N-route Shortest Path Routing

To calculate the shortest, second shortest, and so on, we improve the Dijkstra's algorithm with the following steps. Without loss of generality, let a network have n nodes, with start node 1 and end node n.

Step 1: Calculate all the shortest distances from the start node to each intermediate node with Dijkstra algorithm. We get a sequence of $n-2$ elements from $S(1,2)$ to $S(1,n-1)$.

Step 2: Calculate all the shortest distances from each intermediate node to the end node with Dijkstra algorithm. We get a sequence of $n-2$ elements from $D(2,n)$ to $D(n-1,n)$.

Step 3: Shortest path $L(t)$ passing through intermediate node t will be : $L(t) = S(1,t) + D(t,n)$. $L(2), L(3), \ldots, L(n-1)$, when the sequence is arranged in ascending order, it will result in the shortest path, the second shortest path, the third shortest path and so on.

We use N-route shortest path to transport an XML document efficiently.

3.5.2 N-route Disjointed Shortest Path Routing

In N-route shortest path routing proposed, one node may be used repeatedly in different routes. To avoid repeated use of a node, we introduce N-route Disjointed Shortest Path Routing as follows:

Step 1: Calculate the shortest distances from the start node to target nodes.

Step 2: Delete the nodes from the topology which are capable of XML processing in the shortest route.

Step 3: Calculate the shortest distances from the start node to target nodes as the second route.

Step 4: Repeat the step 2 and 3 to get as many paths as desired.

We use N-route shortest paths to transport XML documents while processing them in the network nodes.

4 EVALUATION EXPERIMENT FOR XML ROUTING

4.1 Experimental Environment

To evaluate the various XML routing algorithms, we use a virtual simulation environment, written in the Scala language. The simulated topology is shown in Figure 2. The shaded nodes are XML processing nodes, node 0 is the start node or source, and node 8 is the terminal node or sink. The number between two nodes represents the distance between them: 1000 distance corresponds to a latency of 10ms. We vary the XML document size (number of tags), document generation frequency and node processing capacity. The minimize hop routing algorithm (MIN) is also simulated to compare its performance with the XML routing algorithms introduced.

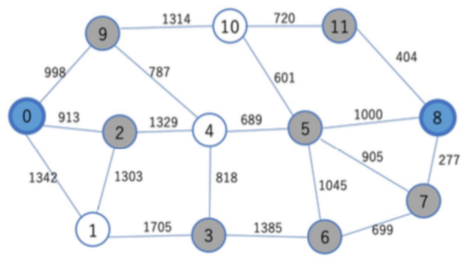


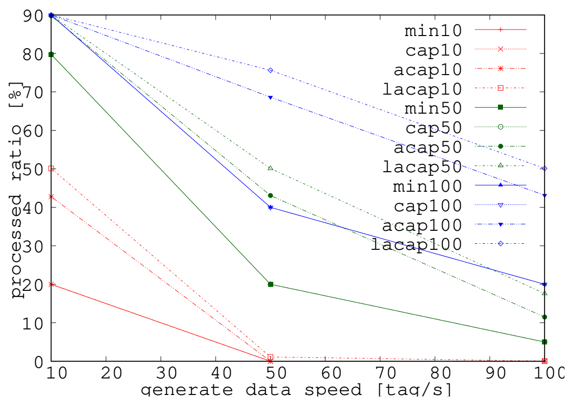
Figure 2: Distributed XML Network Simulation Topology.

4.2 Experimental Results

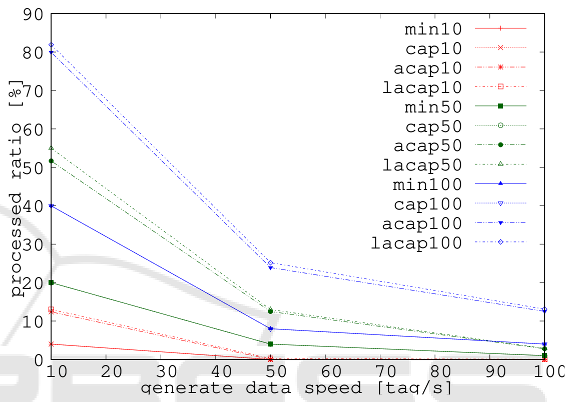
In all figures in this session, the Y axis represents the network processing ratio (network processed tags/total tags) of the system, the X axis is the tag generation speed (tag/s). The title of each sub-graph, such as "document size = 10", denotes the XML document size in number of tags. The labels (min10, cap10, acap10 ...) represent the processing speed of intermediate nodes which are capable of processing and the type of single path computation used: cap10 means capacity routing, and so on. In total, we simulated 36 instances of single path XML routing, 36 instances of two path routing, 36 instances of three path routing, and 36 instances of two disjoint path routing.

4.2.1 Single Path Routing

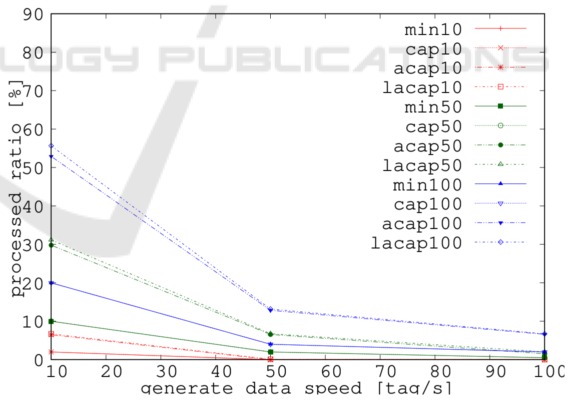
In Figures 3, we first notice that for a same routing algorithm, higher capacity processing nodes lead to higher network processing ratios, as expected, for all document sizes. When comparing different routing algorithms, we find from higher to lower processing ratios *LACAP* > *ACAP* > *CAP* > *MIN*. The results verify that in fact including latency on the routing computation as proposed spreads the XML traffic to more XML processing capable network nodes, delivering best processing ratio performance. Also, we verify that minimum hop routing results in worst performance, because the routes selected use a small number of XML processing capable nodes, and concentrate traffic on the same nodes.



(a) document size = 10.



(b) document size = 50.

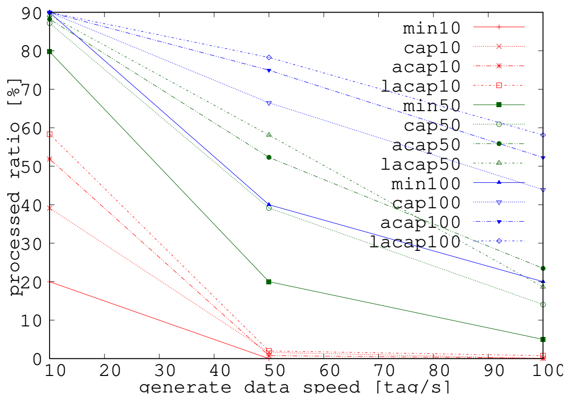


(c) document size = 100.

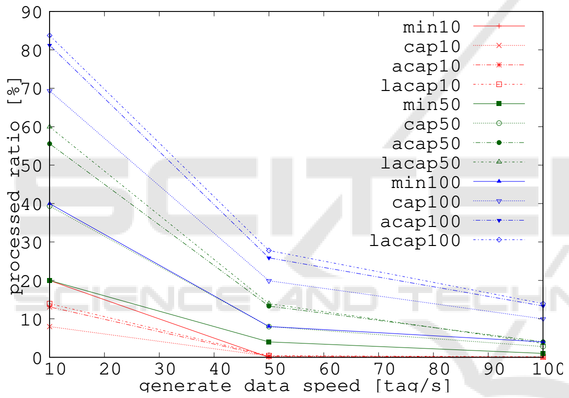
Figure 3: One route shortest path routing.

4.2.2 Multiple Path Routing

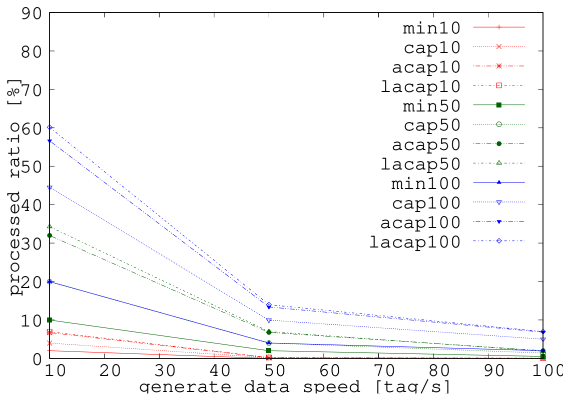
Figures 4 and 5, report on two path XML routing and three path XML routing algorithm results for various document sizes. For small document sizes (Figs. 3 a, b, and Figs. 4 a, b), we see that processing ratios are similar computation strategies with matching weights. For a large document size, 100 tags, we see that in general a two path routing delivers a higher network processing ratio than single route. When comparing two route and three route results, there is no significant improvement for using one extra route, suggesting a saturation point beyond which more routes do not lead to better network XML processing performance.



(a) document size = 10.



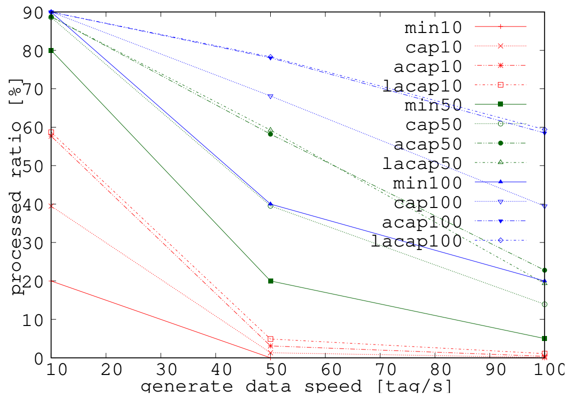
(b) document size = 50.



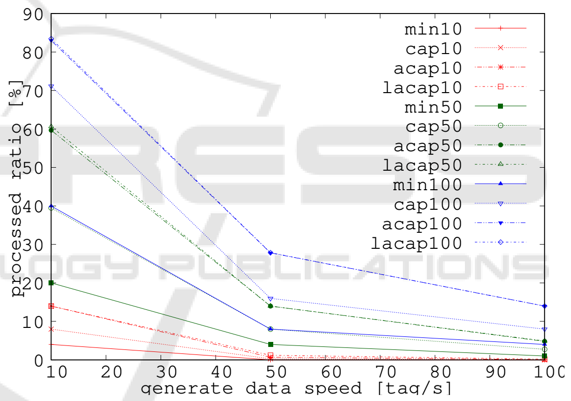
(c) document size = 100.

Figure 4: Two route shortest path routing.

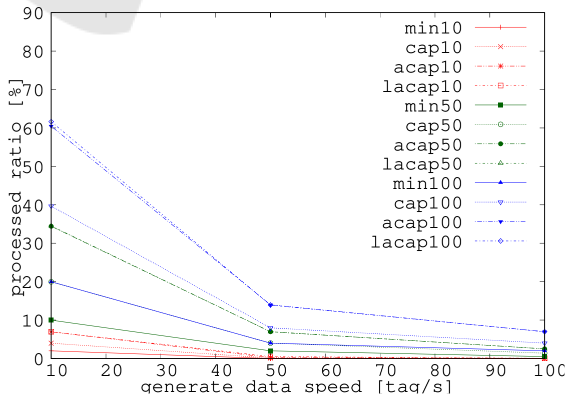
When comparing two path routing with two disjoint path routing (Figs.4 and 6), we can see that two disjoint path routing performs slightly worse than two path with node repetition, especially for ACAP and LACAP weight computation. This is because the weight computation already takes care of previously routed traffic load, so repeating a node will not cause overload processing at a given node. For CAP and MIN hop weight computation strategies, multiple disjoint paths may deliver higher network XML processing performance.



(a) document size = 10.

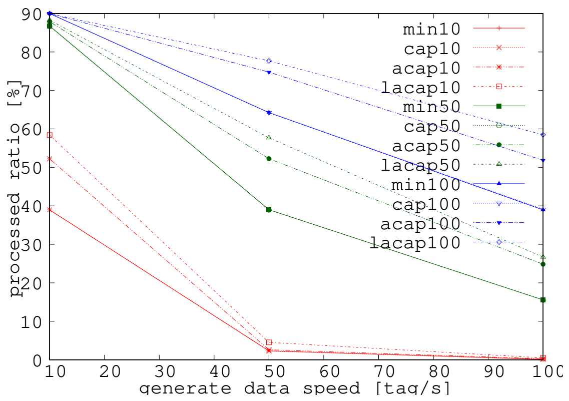


(b) document size = 50.

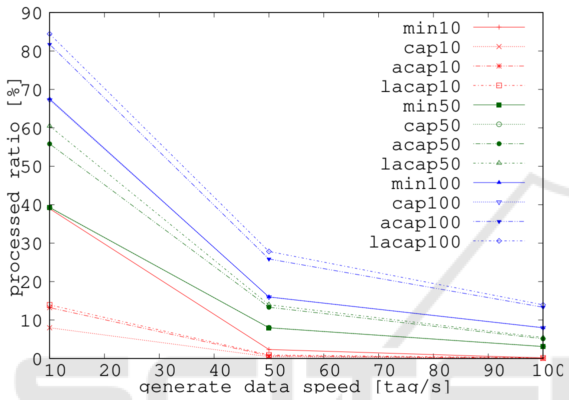


(c) document size = 100.

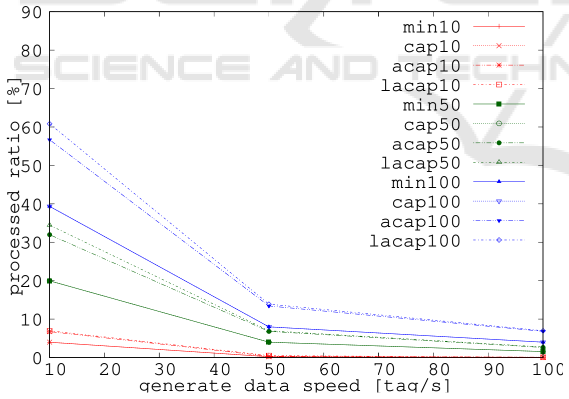
Figure 5: Three route shortest path routing.



(a) document size = 10.



(b) document size = 50.



(c) document size = 100.

Figure 6: Two route disjointed shortest path routing.

5 CONCLUSIONS

In this paper, we have formulated and studied the XML routing problem on a network that contains XML processing capable nodes. We have defined three single path routing strategies, and two types of multiple path routing strategies, covering routing without node repeats, and disjoint multiple paths.

We have evaluated these XML routing strategies on an event driven simulated network, and verified the following behaviors. On single path routing: i) Minimum hop routing delivers poor network XML processing performance; ii) considering available capacity prevents node overload condition; iii) internode latency may be used to spread XML traffic to longer paths, delivering better network processing performance. On multipath routing: i) Two route path delivers better performance for large XML documents; ii) There is a saturation point, beyond which an extra route does not cause performance improvement; iii) Using disjoint paths does not improve XML network processing performance, especially for link costs which already take into account XML node traffic loads.

Directions for future research include multiple path XML routing for fault tolerance.

ACKNOWLEDGEMENTS

This work was supported by Japan and Technology Agency (JST), Strategic International Collaborative Research Program (SICORP), Japan.

REFERENCES

Cavendish, D. and Candan, K. S. (2008). Distributed xml processing: Theory and applications. *Journal of Parallel and Distributed Computing*, 68(8):1054–1069.

Cavendish, D. and Gerla, M. (1998). Internet qos routing using the bellman-ford algorithm. In *Proc. of HPN '98 Proceedings of the IFIP TC-6 Eigth International Conference on High Performance Networking*, pages 627–646.

Fan, Y. and Chen, A. (2012). Energy efficient schemes for accuracy-guaranteed sensor data aggregation using scalable counting. *IEEE Transactions on Knowledge and Data Engineering*, 24(8):1463–1477.

Kalarani, S. and Uma, G. (2013). Improving the efficiency of retrieved result through transparent proxy cache server. In *Proc. of fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT) 2013*, page 1–8.

M. Shimamura, T. I. and Tsuru, M. (2010). Advanced relay nodes for adaptive network services - concept and prototype experiment. In *Proc. of International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA) 2010*, page 701–707, Los Alamitos, CA, USA.

S. H. Kim, K. Kim, C. L. and Ro, W. (2012). Offloading of media transcoding for high-quality multimedia services. *Consumer Electronics, IEEE Transactions on*, 58(2):691–699.

S. Nishimura, M. Shimamura, H. K. and Ikenaga, T. (2012). Transparent caching scheme on advanced relay nodes for streaming services. In *Proc. of International Conference on Information Networking (ICOIN)*, page 404–409.

Solis, I. and Obraczka, K. (2006). In-network aggregation trade-offs for data collection in wireless sensor networks. *Int. J. Sen. Netw.*, 1(3/4):200–212.

Tennenhouse, D. L. and Wetherall, D. J. (2007). Towards an active network architecture. *SIGCOMM Comput. Commun. Rev.*, 37(5):81–94.

Wang, Q. and Ozsu, T. (2007). An efficient eigenvalue-based p2p xml routing framework. In *Proc. of 2007 Seventh Intl. Conference on Peer-to-Peer Computing, P2P '07*, page 105–112.

Y. Kanada, K. S. and Nakao, A. (2012). Network-virtualization nodes that support mutually independent development and evolution of node components. In *Proc. of IEEE International Conference on Communication Systems (ICCS) 2012*, page 363–367.

Y. Uratani, H. Koide, D. C. and Oie, Y. (2012). Distributed xml processing over various topologies: Characterizing xml document processing efficiency. In *Web Information Systems and Technologies, volume 101 of Lecture Notes in Business Information Processing*, page 57–71. Springer Berlin Heidelberg.

Ziyaeva, C. E. G. and Min, D. (2008). Content-based intelligent routing and message processing in enterprise service bus. In *Proc. of 2008 Intl. Conference on Convergence and Hybrid Information Technology, ICHIT '08*, page 245–249.