International Journal of Database Theory and Application
Vol. 2, No. 1, March, 2009

An Efficient Hybrid Path Selection Algorithm for an Integrated Network Environment

P. Calduwel Newton[1], Dr. L. Arockiam[2], Tai-hoon Kim[3]

[1] *Lecturer, Bishop Heber College (Autonomous), Tiruchirappalli - 620 017*
[2] *Lecturer (SG, St.Joseph's College (Autonomous), Tiruchirappalli- 620 002*
[3]*Dept. of Multimedia, Hannam University, Korea*
Tamil Nadu, INDIA
calduwel@yahoo.com

Abstract

The selection of a path in the exponentially growing internet is a difficult task. The upcoming high-speed networks are expected to support a wide variety of communication-intensive real-time multimedia applications. There are many significant factors which affect the performance of a computer network such as choice of routing algorithm, heavy traffic, etc. There must be a mechanism to effectively use the network. Using shortest paths conserves network resources, but it causes some resources of the network to be over utilized while other resources remain under utilized.

As the path selection directly affects the throughput, an efficient path selection algorithm is needed which periodically finds the optimum path. In this paper, we would like propose an algorithm that can be used in both datagram and virtual circuit environment. This is a small step in building throughput-aware and traffic-sensitive path selection. Ultimately, our idea contributes to find a route in an integrated network environment.

Keywords : *Heterogeneous Data, Optimum Path, Throughput, Bandwidth, Network Optimization, Traffic*

1. Introduction

Today's network faces many challenges to carry heterogeneous data. There are many paths that are potential in the Internet connecting any two hosts. [1] Some of these paths have higher bandwidth than others, some have lower propagation delay, and others see less congestion. These factors are responsible for the end-to-end performance achievable along any given path.

Path Selection Algorithms should provide good end-to-end performance, balance load, minimize cost, etc. It must possess the knowledge about what the best hop is.

This paper proposes such a path selection algorithm in which periodic i-paths selection is done. The remaining part of this paper is organized as follows. Section 2 provides an introduction to Network and path selection. Section 3 analyses the related work on path selection algorithms. Section 4 explains some of the existing path selection algorithms and its demerits. Section 5 describes the proposed algorithm. Section 6 contributes to the conclusion and future enhancements of this paper. Finally, Section 7 reveals the references.

2. Network and path selection

A network N is represented as an undirected graph (V, E), where the node set V represents the set of computers in the network, and the edge set E represents the set of communication links in the network. For simplicity, we assume that N is connected. A *path* in V is a sequence of nodes from V. A *network path* P in N is a sequence of nodes in V where each pair of consecutive nodes in P is an edge in E. We denote by *head(P)* the first node in the path of P, and by *body(P)* the nodes remaining in P after removing its first node.

[2] A path P in network N is *rooted* iff the last node in P is the root node, i.e., P = v_k, v_{k-1}, v_0 where v_0 = root. Let v = head(P), and assume node v has chosen P as its path to the root. Then, node v routes data messages to its neighbor w where w is the next node after v in P.

We assume each node v has a set of paths, denoted by Sel(v), which are acceptable to v. That is, the path chosen from v to the root *must* be chosen from this set.

More formally, a *path selection* is a pair (V, Sel), where V is a set of nodes, with a distinguished root node and Sel() is a function, which takes a node v, $v \in V - \{root\}$, and returns a non-empty set of rooted paths over

V, whose initial node is v. Without loss of generality, we assume $Sel(v) \neq \emptyset$ for all v.

If node v routes its data messages towards its neighbor w, then the remainder of the path which is followed by the data messages from v depends on the next hop choice made by w, and so on. This choice of next hop neighbors must be consistent to avoid undesirable conditions such as loops. We define this consistency formally as follows.

A rooted path P is *selectable* in (V, Sel) iff $P \in Sel(v)$, where v = head(P). A path selection (V, Sel) is *satisfiable* in network N, N = (V, E), iff there exists a spanning tree T of N, such that, for any v, $v \in V$, the rooted path from v to the root node along T is a selectable path in (V, Sel). We refer to this spanning tree as a *satisfiability tree* of (V, Sel) in N.

Thus, each node v should choose as its next hop neighbor its parent in T. This guarantees that the path followed by the data messages of v is a selectable path. However, a path selection may not be satisfiable in some networks, and thus, it may not have a satisfiability tree.

This discussion given above describes about the nodes and edges in a network. Here node and edge refers to the host and link, respectively. It shows that many paths are available in a network. Hence path selection is vital and path selection algorithms could not be avoided.

3. Related research work

This section shows the study of various literatures about the behavior of existing path selection algorithms. [3] Paxson provides an end-to-end study of path dynamics by using the traceroute tool to identify the particular hops traversed between pairs of hosts. He finds that a single route generally dominates Internet paths, but that some networks do experience significant route fluctuation. Moreover, his data indicates that a large and increasing fraction of Internet paths follow different routes from source to destination than from destination to source.

Another area in which there is significant literature is the blackbox study of Internet path characteristics. [4] Bolot uses Internet Control Message Protocol (ICMP) “echo” packets to examine the distributions of packet loss and round-trip times observed on a single trans-Atlantic path.

[5] A recent study by Paxson examines the characteristics of a larger set of paths using an automated analysis of Transmission Control Protocol (TCP) data transfers. Paxson's results indicate that there is a wide variation in path characteristics such as round-time, packet loss, and bandwidth. However, he also finds that the amount of available bandwidth tends to be stable for time periods up to several hours.

[6] Balakrishnan et al. also find significant temporal stability in bandwidth measurements collected from the IBM Olympic Web servers. Further, they show that hosts which share portions of a path tend to obtain similar amounts of bandwidth.

[7] Varadhan et al. present a simulation study about the effect of path changes on the performance of transport protocols. They show that small path changes during a TCP session can lead to significant reordering and a consequent reduction in performance. Connectionless network uses multiple routes to send the packets and achieves higher bandwidth [8].

Finally, [9] Francis et al. explore the possibility of using end-to-end measurements to construct maps of the minimum Internet propagation delay between hosts. Their methodology is to predict the minimum propagation delay between a pair of hosts by triangulation using a series of pair-wise measurements.

4. Existing path selection algorithms

This section identifies various existing algorithms and explains demerits of these algorithms. There are many path selection algorithms, which are used to send the data efficiently. In the path selection process, a number of factors need to be considered. Some of them are as follows.

Path length: Quality of Service (QoS) routing algorithms that favor shorter paths yield better routing performance. This factor implies that the shortest path algorithm is a good algorithm to select the candidate paths.

Load balancing: Since the set of candidate paths is predefined, carefully selecting the paths to evenly distribute the load will result in good performance. Load balancing must be done in the global scale, that is, the aggregate effect of all candidate paths on all links in the network must be considered.

Shared links: For a given source-destination pair, multiple candidate paths are to be selected. We should minimize the number of shared links in the candidate paths for a given source destination pair so that path selection can be adaptive to the network state. Essentially, when a link is under heavy load, algorithm should select other paths that do not use the link for connection requests.

An efficient path selection scheme should optimize all these factors. In the following, [10] we will present a number of path selection methods that are used today.

4.1. Widest-shortest path

Widest-shortest path selects a path with minimum hop count among all feasible paths. In this case, higher bandwidth cannot be achieved. [11] If there are several such paths, the one with the maximum reserved bandwidth is selected.

4.2. Shortest-widest path

Shortest-widest path selects a path with the maximum bandwidth among all feasible paths. In this case distance may be larger. [10] If there are several such paths one with the minimum hop count is selected.

4.3. Breadth first search path selection

This method focuses on selecting shortest paths between as candidate paths. For each source-destination pair, the method uses a breadth first search algorithm to find all minimum-

hop paths, as well as some alternative paths with a limited extra path length. It then selects from the found paths the set of candidate paths. This method ignores the global load balancing and shared link issues.

4.4. Per-pair path selection

This method attempts to find shortest paths between each source-destination pair while minimizing the number of shared links in the candidate paths for a given source-destination pair. This method selects candidate paths for each source destination pair in the following manner. It initially assigns the same weight, 1, to all links in the network and use the Dijkstra shortest path algorithm to find the first candidate path. After the method finds a candidate path, it increases the weights on all the links along the path and then repeats the process until the number of candidate paths reaches the target or no more new paths can be found. By increasing the weights for the links in the selected paths, the method tends to avoid using the links that are in the selected candidate paths and thus, minimize the number of shared links in the candidate paths. However, this method does not consider the global load balancing issue.

4.5. Global path selection

Global Path selects paths for all source-destination pairs such that the load is evenly distributed to all links. This is done as follows:

Global Path first assigns the same weight, 1, to all links in the network. It then considers each pair of nodes in a round-robin fashion. For the first pair of nodes, it selects one shortest path using the Dijkstra shortest path algorithm and increases the weights on the links along the path. It does the same for the second source destination pair and so on. Since the weights of the links on the paths chosen previously have been increased, it is likely that the links in the selected paths will not be chosen in the future. Since Global Path considers all source-destination pairs, it is likely that the load on all links will be balanced at the end of the path selection process.

5. Proposed algorithm

This section proposes an algorithm and describes the algorithm with an example. The steps are as follows. Our main aim in proposing this algorithm is to save time and to reduce congestion.

```
Start
Bandwidth request ‘r’ arrives at time ‘i’ to send data
                between source ‘a’ and destination ‘e’
Estimate all ‘n’ available paths

  If no path available
  then
    reject the request
        endif

  repeat until all paths are identified
          then
       store path information
```

end repeat

repeat until all paths are computed
then
compute the available bandwidth for all paths
compute the time required for all paths
end repeat

Select the paths that require minimum time
Stop

This algorithm estimates all available paths that can be used for the data transfer after receiving the bandwidth request from any host. At times, there may be no path available for data transfer. In such case, the request will be rejected. Obviously, it leads the network into congestion. If path(s) exist then store all information about each path. After storing all the paths, available bandwidth is calculated. With the help of available bandwidth, time required for each path to transfer data is computed.

Using this information, paths that require minimum time to transfer data is considered as i-paths. These i-paths give higher throughput and reduces congestion. As network is growing exponentially, paths selected as intelligent may not be permanent. They will be changed quickly. So, the i-paths should be selected periodically. This algorithm can be executed periodically in order to keep the latest information.

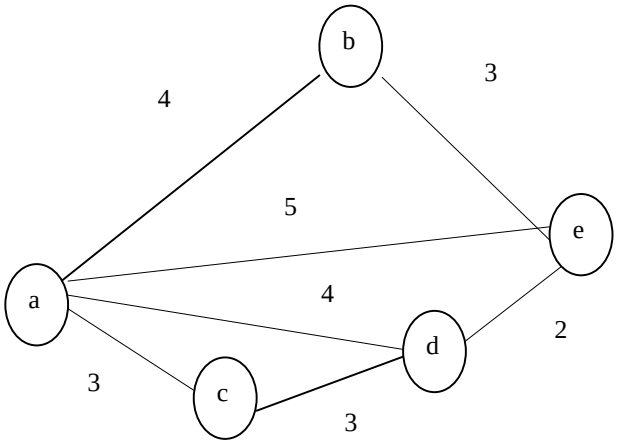

Figure 1. A Network

In Figure 1, the alphabet in the node indicates the host name and number on the edge indicates the distance. [12, 13] By keeping track of each flow, nodes that could not provide requirements and nodes that are misbehaving can be identified and corrected [14]. The following are assumptions made in this network.

Assumptions (Datagram environment)

Bandwidth of all the links : 10 Mbps

Data to be sent : 8.5 Mb

According to Figure 1, the following table gives information about various paths.

Table 1. Details About Various Paths

S.NO.	PATH	PATH INFORMATION
1.	Path 1	a-b-e
2.	Path 2	a-e
3.	Path 3	a-d-e
4.	Path 4	a-c-d-e

Here, we would like to explain how the proposed algorithm works with an example data. First, this algorithm identifies number of available paths in the network. Column 2 and 3 of Table 1 shows that the network has four paths and full details about all the paths, respectively. Second, it computes the available bandwidth for all the identified paths. The following is an assumption about available bandwidth.

Path 1 – 50%
Path 2 – 5%
Path 3 – 30%
Path 4 – 75%

The above information shows that shortest path (Path 2) is over utilized. But, the longest paths (Path 1& Path 4) are under utilized. As per the Figure 1, the time required to send the data are as follows.

Path 1 – 7 secs.
Path 2 – 5 secs.
Path 3 – 6 secs.
Path 4 – 8 secs.

If the packets are sent in all the paths (ie. from Path 1 to Path 4), they take 8 secs. to reach the destination. Assume that path 1 sends 2 Mb, path 2 sends 0.5 Mb, path 3 sends 3 Mb and path 4 sends 3 Mb. If the proposed algorithm is used, the above packets will take only 7 secs. to reach the destination. The explanation is as follows. The available bandwidth for path 1 is 50%. That means, it can send 5 Mb of data in single transfer. Likewise, path 2 can send 0.5 Mb of data and path 3 can send 3 Mb of data. So, it only takes 7 secs. and also available bandwidth for path 4 can be used by any other data transfer. Instead of using all the paths, selected paths are used to increase the throughput.

Assumptions (Virtual-circuit environment)

Bandwidth of all the links : 10 Mbps
Data to be sent : 5 Mb

In Virtual-circuit environmet the first and second steps are same as the Datagram environment. Third, it computes the time required to send the data. The computational results are:

Path 1 – 7 secs.
Path 2 – 50 secs.
Path 3 – 12 secs.
Path 4 – 8 secs.

The Fig.1 shows that Path 2 is the shortest path. But it cannot be used to send the data as it is heavily loaded. So, Path 1 is selected to send the data. Though it is longer than Path 2, it only takes minimum time. After some time, Path 1 may take more time than Path 2, Path 3 and Path 4. If the proposed algorithm is used, it will be executed periodically and finds the optimum path dynamically.

6. Conclusion and Future enhancements

In this section, we would like to suggest the advantages, disadvantages and future enhancements of our proposed algorithm. It can be used by both Datagram (Connectionless) and Virtual-circuit (Connection-oriented) environment. In Datagram environment, it identifies only the intelligent paths instead of multiple paths. These intelligent paths contribute significantly to improve the performance of the network. Ultimately, it helps to save time, increase throughput, and reduce congestion.

The weakness of this algorithm is computational overhead. Measures can be taken to reduce the above weaknesses. As a future enhancement, this algorithm could be implemented and extended for wireless networks. This is a small step in providing an efficient path selection algorithm for an integrated environment.

Acknowledgement

This work was supported by a grant from Security Engineering Research Center of Ministry of Knowledge Economy, Korea.

7. References

[1] Stefan Savage, Andy Collins, Eric Hoffman, John Snell and Thomas Anderson, “The End-to-End Effects of Internet Path Selection”, SIGCOMM 1999: pp. 289-299.
[2] Jorge Cobb, ``On The Selection of Optimum Paths in Computer Networks", IEEE International Conference on Communications (ICC), June 2001.
[3] Vern Paxson, “End-to-End Routing Behavior in the Internet”. In Proceedings of the ACM SIGCOMM '96, Stanford, CA, August 1996, pp. 25–38.
[4] J-C. Bolot, “End-to-end Packet Delay and Loss Behavior in the Internet”. In Proceedings of the ACM SIGCOMM '94, San Francisco, CA, September 1993, pp. 289–298.
[5] Vern Paxson, “End-to-end Internet Packet Dynamics”. In Proceedings of the ACM SIGCOMM '97, Cannes, France, September 1997, pp. 139–152.
[6] Hari Balakrishnan, Srinivasan Seshan, Mark Stemm, and Randy H. Katz, “Analyzing Stability in Wide-Area Network Performance”. In Proceedings of the 1997 ACM SIGMETRICS Conference, Seattle, WA, June 1997.
[7] Kannan Varadhan, Deborah Estrin, and Sally Floyd, “Impact of Network Dynamics on End-to-End Protocols: Case Studies in TCP and Reliable Multicast”. Technical Report USC-CS-TR 98-672, University of Southern California, Information Sciences Institute, April 1998.
[8] Andrew S. Tanenbaum, “Computer Networks”, Fourth Edition, 2005, pp.423.

[9] Paul Francis, Sugih Jamin, Vern Paxson, Lixia Zhang, Daniel Gryniewicz, and Yixin Jin, "An Architecture for a Global Internet Host Distance Estimation Service". In Proceedings of the IEEE INFOCOM '99, New York, NY, March 1999.
[10] E.George Dharma Prakash Raj, S.V.Kasmir Raja, Sinthu Janita Prakash, "QoS Routing for Communication Quality" Proceedings of the Asia Pacific Conference on Parallel and Distributed Computing Technologies, Vellore Institute of Technology, Vellore. South India, December 2004, pp. 566 – 573.
[11] Shyam Subramanian and Venkatesan Muthukumar, "Alternate Path Routing Algorithm for Traffic Engineering", ITCC 2003: pp. 367 Electronic Edition.
[12] Calduwel Newton P , George Dharma Prakash Raj, Thomson Fredrick "An Adaptive Throughput-Aware Route Selection Algorithm", Proceedings of the International Conference on Advanced Technologies in Telecommunication and Control Engineering, INTI College , Malaysia, August 2006, pp.66.
[13] Calduwel Newton P., "A Contemporary Technique to Guarantee Quality of Service (QoS) for Heterogeneous Data Traffic", Proceedings of the International Conference on Information Security and Assurance, IEEE CS, April 2008. Korea, pp. 210 – 213.
[14] Calduwel Newton P., "A Time-Sensitive Path Selection Algorithm for Connectionless Network", Proceedings of International Symposium on Ubiquitous Multimedia Computing, IEEE CS, Australia, October 2008, pp. 79-82.