USING MAXIMUM AND MAXIMUM CONCURRENT MULTICOMMODITY FLOWS FOR QOS ROUTING IN THE INTERNET

By

SATOKO CHIKA

Bachelor of Science

The University of Aizu

Aizu-Wakamatsu, Japan

2002

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2007

USING MAXIMUM AND MAXIMUM CONCURRENT

MULTICOMMODITY FLOWS FOR QOS

ROUTING IN THE INTERNET

Thesis Approved:

Dr. Venkatesh Sarangan

Thesis Adviser

Dr. John P. Chandler

Dr. Nohpill Park

Dr. A. Gordon Emslie

Dean of the Graduate College

Acknowledgments

I would like to express my deep gratitude to my thesis advisor Dr. Sarangan. I could not complete this thesis without his advice, strong support, and earnest labor. I am also thankful to committee members, Dr. Chandler and Dr. Park. Further, I would like to thank Dr. Chandler for his advice on statistic areas and support throughout this thesis.

I am also thankful to the Computer Science Department for giving me a teaching assistant position. I learned many things from both supervisors and students. These wonderful experiences will help my future work.

I would like to convey my gratitude to my friends for supporting me. Warm and kind words of encouragement pushed me forward beyond any setbacks.

Finally, I would like to express my sincere thanks to my parents and sister. I could not overcome life abroad without their strong support and encouragement.

TABLE OF CONTENTS

IV. FINDINGS

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

The Internet is a major source of information, news and entertainment. “Connecting” was the highest priority when the Internet was designed and implemented. Now, there is no question that the Internet is “connected”. The Internet has started using Quality of Service (QoS) features, and QoS is the key for the further success of the Internet. With a variety of services over the large network, users expect QoS improvements.

An inevitable issue in QoS routing is scalability. Frequently updating state information is required in QoS routing, and it causes not only consumption of network bandwidth but power to generate and advertise update messages [1]. This scalability concern increases as the network size becomes larger.

Hierarchical QoS routing is one of the solutions to deal with this scalability concern in a larger network. In hierarchical routing, nodes are grouped and treated as domains. This clustering is repeated to create a multi-level hierarchy. Since continually updating detailed information at every router increases the communication overhead in such a large network, topology information is aggregated before being advertised. This technique is called topology aggregation [2]. In hierarchical QoS routing with topology

aggregation, upper level nodes receive concise topology information from lower level nodes [12].

A general maximum flow algorithm is used to find a feasible path in bandwidth aggregation. It displays a high Bandwidth Over-estimation Ratio (BOR) and low Bandwidth Under-estimation Ratio (BUR). The maximum multicommodity flow and maximum concurrent flow algorithms can reduce BOR because of their more accurate acceptance criteria. The objective in this thesis is to compare the statistical results obtained by simulation using general maximum flow algorithm and maximum multicommodity flow and maximum concurrent flow algorithms.

CHAPTER II

REVIEW OF LITERATURE

2.1 Quality of Service (QoS)

Assurance of QoS involves selecting a path with sufficient resources for the requested QoS parameters, such as bandwidth, delay, jitter and packet loss. The QoS problem is intractable when dealing with multiple constraints [7].

Many applications use the Internet, and packets flowing within the network include those for Web browsers, e-mail and so on. The sender has to deal with routing of various commodities. On the other hand, the receiver expects data to be delivered with guaranteed QoS. Frequent updating is needed in QoS routing. However, as the network becomes larger, it is almost impossible to advertise topology information to every node in the network due to the time, space and bandwidth required. Therefore, scalability will be the main concern as network size increases. There are two possible solutions to deal with this problem: quantity reduction and frequency reduction [13]. Quantity reduction emphasizes reducing the number and size of the messages which must be sent to inform one router of another's state. The objective of frequency reduction is to try to create messages infrequently as keeping routing performance. The topology aggregation scheme belongs in the quantity reduction strategies [1].

2.2 Hierarchical Routing Algorithm

The nodes are classified into groups, and nodes and links in a group are aggregated recursively [11]. Figure 1 (a) shows a physical network. A.a.*, A.b.*, A.c.*, B.a.* and so on are clustered and represented as a logical node, A.a, A.b, A.c and B.a and so on respectively. This is first-level abstraction and shown in Figure 1 (c). And then repeatedly A.* is clustered to A, B.* is clustered to B, and C.* is clustered to C. This is second-level abstraction and shown in Figure 1 (d).

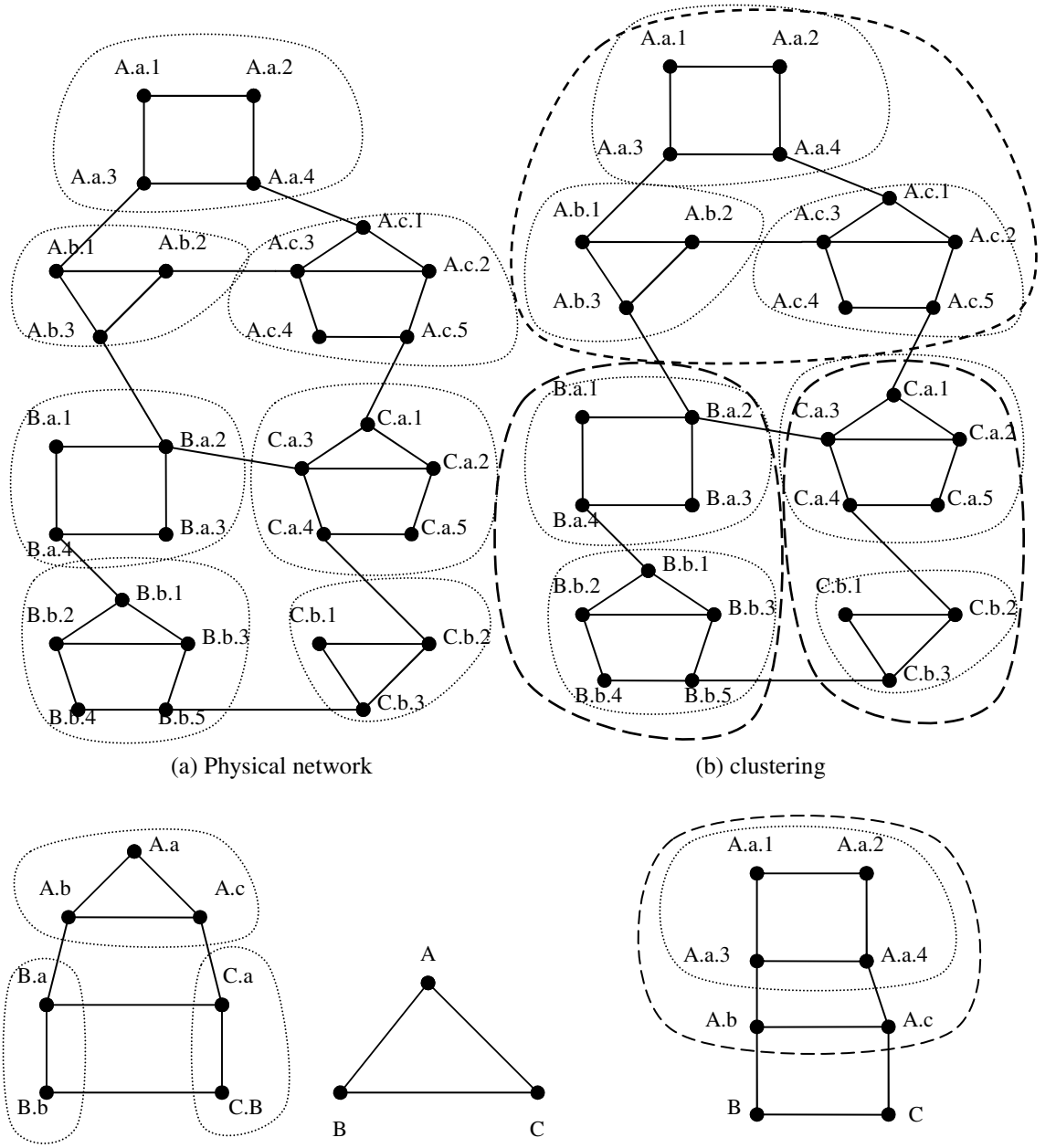
2.3 Topology Aggregation (Network State Aggregation)

The objective of the topology aggregation technique is to allow the application of both concise and detailed information to the routing. This topology aggregation is important for scalability in hierarchical large networks and contributes to the reduced overhead. The nodes are clustered into domains and each domain is connected with others through border routers (BRs). BRs in a particular domain have detailed information about their own domain and concise information about neighboring domains. For example, A.a.3 in Figure 1 (e) has detailed information about connectivity and resources of its domain (A.a.1, A.a.2, and A.a.4) and aggregated information of A.b. A BR sends reduced network information to nodes outside of its domain and detailed information to those within its domain. In other words, it has a complete view for inside

the domain and aggregated view for outside the domain. Detailed inside domain information is aggregated before being advertising. The exchanged information between domains contains an aggregation of the connectivity and resource availability [10]. Hierarchical QoS routing is the technique to find a feasible path using these detailed and aggregated information sets [9].

2.4 Maximum Flow Problem

The objective of the maximum flow problem is to find a flow of maximum value from a *single*-source to a *single*-sink in a flow network $G = (V, E)$, where V is a vertex set and E is a collection of edges. Ford-Fulkerson, Edmonds-Karp, and push-relabel algorithms are major algorithms to solve maximum flow problem [8]. In this simulation, Edmonds-Karp is used. We add a super-sink to calculate the advertised capacity. Figure 2 shows an example of calculating advertised capacity when domain B requests to domain A. The edges are added from c, d, and e to the super sink with an infinity capacity. Adding a super sink enables max flow algorithm to calculate advertised capacity of the topology with multiple sinks.



(a) Physical network (b) clustering

(c) First-level abstraction (d) Second-level abstraction (e) the network image viewed by node A.a.1

Figure 1. Hierarchical Network Model [7]

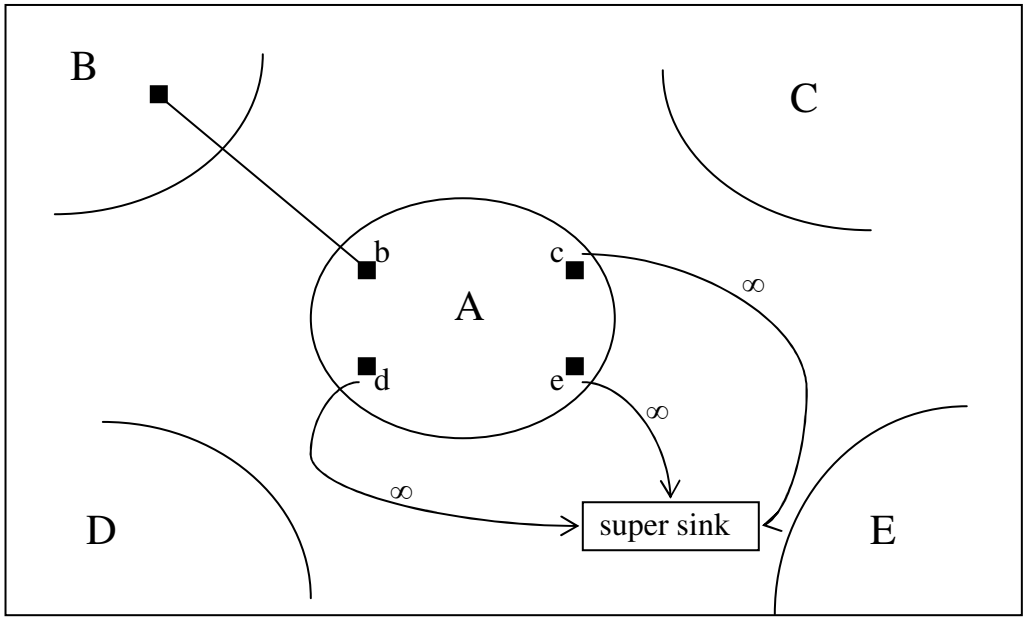


Figure 2. Max Flow with Multiple Sinks

2.5 Multicommodity Flow Problem

Given a graph $G = (V, E)$ with a non-negative capacity $u \geq 0$. There are k commodity pairs $(s_1,t_1),(s_2,t_2),.....(s_k,t_k)$. s_j and t_j represent source and sink of commodity j, $1 \leq j \leq k$ respectively. Let f_i be a real-valued function. $f_j(u,v)$ is the flow from vertex u to v of commodity j. Define $f(u,v)$ as an aggregate flow. The aggregate flow on edge (u,v) is sum of all commodity flows, $f(u,v) = \sum_{j=1}^{k} f_j(u,v)$ and cannot exceed the capacity of edge (u,v). The multicommodity flow problem is to determine if there is such a flow.

2.6 Maximum Multicommodity Flow Problem

Given a directed network $G = (V, E)$ with non-negative capacities u, E -> **R** and k ordered pairs of vertices $(s_1, t_1), (s_2, t_2), ...(s_k, t_k)$. The pair, (s_j, t_j), $1 \leq j \leq k$ represents a source-sink terminal pair for commodity j. Let $\left|f_j\right|$ be the amount of flow from s_j to t_j. The objective of maximum multicommodity flow problem is to maximize the sum of flows, $\max \sum_j \left|f_j\right|$ without exceeding capacity of any edge [3]. Let P_j denote the set of paths from s_j to t_j and let $\mathrm{P} := \cup_j \mathrm{P}_j$. Let $x(P)$ define as an amount of flow sent along path P and let $u(e)$ be a capacity of edge e. The linear programming formulation is defined as follows:

$$\max \sum_{P \in \mathrm{P}} x(P)$$
$$\forall e : \sum_{P : e \in P} x(P) \leq u(e)$$
$$\forall P : x(P) \geq 0.$$

The length of an edge means the "marginal cost of using an additional unit of capacity of the edges" [3]. Let l denote the length function. When applying lengths to each edge in the graph, the dual for the above linear program is given as follows:

$$\min \sum_e u(e)l(e)$$
$$\forall P : \sum_{e \in P} l(e) \geq 1$$
$$\forall e : l(e) \geq 0$$

Fleischer [3] provides pseudo-code for maximum multicommodity flow problem invented by Garg and Konemann [6], and it is shown in Figure 3. This pseudo-code is used for simulation.

Input: network G, capacities $u(e)$, commodity pairs (s_j, t_j), $1 \leq j \leq k$,
accuracy ε
Output: primal (infeasible) and dual solutions x and l

Initialize $l(e) = \delta \quad \forall e, x \equiv 0$
while there is a $P \in \mathrm{P}$ with $l(P) < 1$
select a $P \in \mathrm{P}$ with $l(P) < 1$
$u \leftarrow \min_{e \in P} u(e)$
$x(P) \leftarrow x(P) + u$
$\forall e \in P, l(e) \leftarrow l(e)(1 + \frac{\varepsilon u}{u(e)}$
end while
Return (*x, l*).

Figure 3. Maximum Multicommodity Flow Pseudo-Code

2.7 Maximum Concurrent Flow Problem

Given a directed network $G = (V, E)$ with non-negative capacities u, E -> **R** and k ordered pairs of vertices $(s_1, t_1), (s_2, t_2), ...(s_k, t_k)$. At commodity j, source-sink pair, (s_j, t_j) has demands d_j, $1 \le j \le k$. The objective of this problem is to maximize the possible portion of all demands: $\max \lambda, \left| f_j \right| \ge \lambda d_j, \forall j$. Let P_j denote the set of paths from s_j to t_j and let $\mathrm{P} := \cup_j \mathrm{P}_j$. $x(P)$ is defined as an amount of flow sent along path P and $u(e)$ denotes a capacity of edge e. The notation z_j stands for weights at commodity j. The maximum concurrent flow problem is represented using linear program as follows.

$$
\begin{aligned}
&\max \lambda \\
&\forall e : \sum_{P: e \in P} x(P) \le u(e) \\
&\forall j : \sum_{P \in \mathrm{P}_j} x(P) \ge \lambda d_j \\
&\forall P : x(P) \ge 0
\end{aligned}
$$

The lengths are assigned to the edges just as in the maximum commodity flow problem. In addition to them, weights are assigned to the commodities. The length of an edge means “the marginal cost of using an additional unit of capacity of the edges” [3]. The weight is defined as “the marginal cost of not satisfying another unit of demand of the

commodity" [3]. Let l denote the length function. The dual for above linear program is expressed as follows:

$$\min \sum_e u(e)l(e)$$
$$\forall j, \forall P \in \mathrm{P}_j : \sum_{e \in P} l(e) \geq z_j$$
$$\sum_{1 \leq j \leq k} d_j z_j \geq 1$$
$$\forall e : l(e) \geq 0$$
$$\forall j : z_j \geq 0$$

Fleischer's paper [3] introduces maximum concurrent flow pseudo-code in Figure 4. This simulation is based on this pseudo-code. Demands should be given as we can see from her pseudo-code; however, they are not provided in the network topology we use in this simulation. They are calculated as follows. Add a super sink to border routers which are not the source border router. Then, calculate maximum flow from source to the super sink and divide it by the number of (border routers -1). 1 is for source border router.

Input: network G, capacities $u(e)$, vertex pairs (s_j, t_j) with demands $d_i, 1 \le i \le k$, accuracy ε

Output: primal (infeasible) and dual solutions x and l

Initialize $l(e) = \delta / u(e), \forall e, x \equiv 0$

while $D(l) < 1$

for j = 1 **to** k do

$d'_j \leftarrow d_j$

while $D(l) < 1$ and $d'_j > 0$

$P \leftarrow$ shortest path in P_j using l

$u \leftarrow \min\{d'_j, \min_{e \in P} u(e)\}$

$d'_j \leftarrow d'_j - u$

$x(P) \leftarrow x(P) + u$

$\forall e \in P, l(e) \leftarrow l(e)\left(1 + \frac{\varepsilon u}{u(e)}\right)$

end while

end while

Return (x,l)

Figure 4. Maximum Concurrent Flow Pseudo-Code

2.8 Goodness Measures

The following are used for statistical measurements [10].

- Bandwidth Admission Ratio (BAR): Ratio of bandwidth admitted to bandwidth requested
- Bandwidth Over-estimation Ratio (BOR): Ratio of bandwidth dropped in a domain to bandwidth sent
- Bandwidth Under-estimation Ratio (BUR): Ratio of non-forwarded bandwidth that could have been successful to bandwidth not sent

2.9 General Maximum Flow vs. Maximum and Maximum Concurrent Multicommodity Flow Algorithms

Suppose that a border router in domain B wants to send to domain E through domain A in Figure 5. In the general maximum flow problem, it is assumed that other neighbor domains of A, C and D are not using resources in A. In other words, domains C and D cannot share resources in domain A. This makes BOR increase and BUR decrease. We believe that maximum multicommodity flow and maximum concurrent flow algorithms can reduce BOR since their property allows sharing resources among domains.

The maximum multicommodity flow and maximum concurrent flow algorithms can be better than the general maximum flow algorithm.

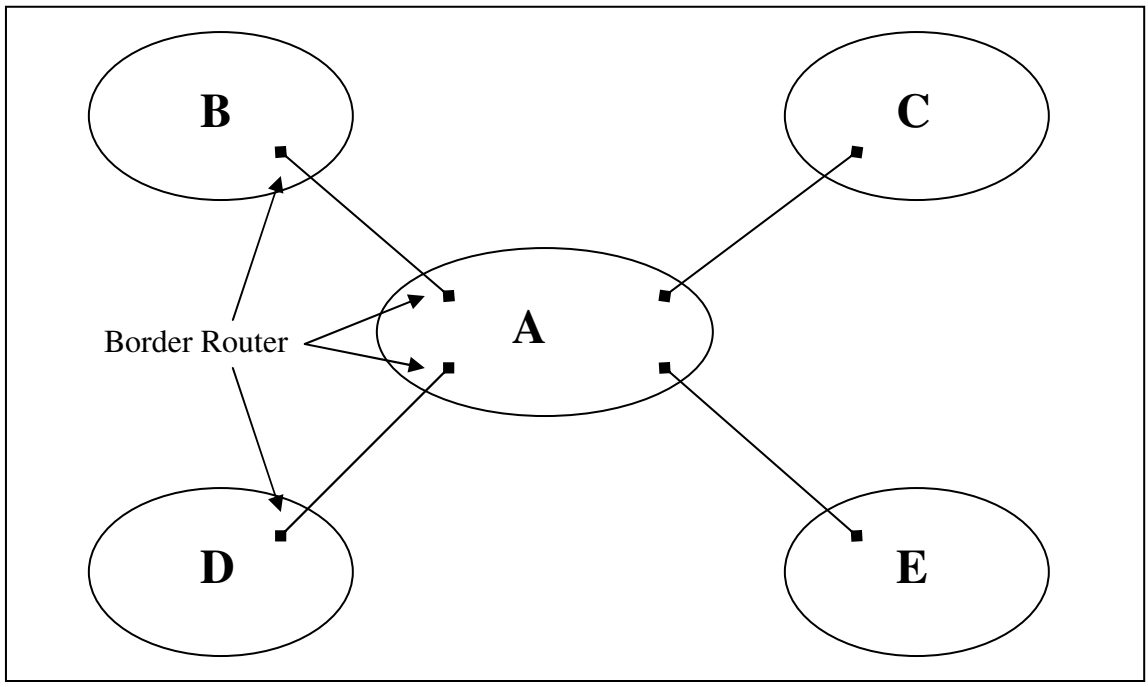


Figure 5. Network Model

CHAPTER III

METHODOLOGY

3.1 Discrete Events Simulator (DES)

A DES is a dynamic system model to simulate the flow of people, things or events, where the system consists of queued events and a virtual clock. Simulation languages such as GPSS, SIMSCRIPT and SIMULA, as well as simulation software have been developed. Programming languages can substitute for simulation languages or simulation software. In this research, we used the Java language to implement both a DES as well as the routing algorithms to be applied.

3.2 Implementing a DES

The system is simulated for a given number of events. Events can be one of two kinds: (1) resource reservation event or (2) resource release event. A new, resource reservation event includes four pieces of information: the time the event occurred, source, destination and bandwidth required to allocate. The time which the event occurred is generated by the equation, $T_n = T_{n-1} +$ (exponential random value, μ_1), $1 \le n \le$ maximum number of events. Source and destination domain are randomly chosen from neighbors of

main domain and created such that source and destination are not equal. Neighbor domains request a bandwidth amount of either 1 or 2. The system updates advertised capacity every 30 seconds as a default value. Once an algorithm decides that the requested bandwidth can be allocated, then the time when resource along the selected path will be released is randomly generated by the equation, $releaseTime_n = T_n +$ (exponential random value, μ_2), $1 \le n \le$ maximum number of events. This release event is added to a heap data structure. The current network topology, G is copied and edges whose capacity is less than requested bandwidth are dropped. This copied topology, G' is used by Edmonds-Karp algorithm to see if a feasible path is found in the Network topology. A code sequence of the DES, when the number of events is variable, and the values of average time between requests and time between updates are fixed, is summarized in Figure 6.

3.3 Assumptions

Here is a summary of the assumptions made in this simulation.

1. If the network cannot reserve a requested unit, the request is dropped and has no second try.
2. Source and destination nodes are not the same.
3. Requested bandwidth is either 1 or 2.

4. All edges' capacity is a fixed number, 4.
5. For more accurate results, the simulation is taken 50 times and then averaged.
6. The default values for maximum number of events, update time, and the average time between requests are 20000, 500 seconds, and 0.05 seconds respectively.
7. The average time a request uses resource in the network is fixed at 10 seconds.

```
for i from 1 to MaxEvents do{
        // generate event (time)

        // do resource release events

        // if current time modulo 500 == 0, update advertised capacity

        // generate event (source)
        // generate event (destination)
        // generate event (bandwidth, 1 or 2)

        // check advertised capacity
        // copy topology G' ← G
        // drop edges whose capacity are less than requested bandwidth in G'
        // find a path using Dijkstra

        If (advertised capacity > request){
                If (path is found in G'){
                        // generate resource release event time
                        // add this event to heap
                }
                else if (path is NOT found in G')
                        // over-estimation
        }
        else if (advertised capacity ≤ request){
                if( path is found in G')
                        // under-estimation
        }
}
```

Figure 6. Discrete Events Simulator

3.4 Input

The DES program takes one input file as an argument that contains the network topology of the main domain with adjacent matrix style: 0 represents no edges between vertices and 1 means there is an edge between vertices.

3.5 Simulation Circumstances

In this simulation, we tested three circumstances as follows [10].

1) BAR/BOR/BUR vs. Number of events
2) BAR/BOR/BUR vs. Average time between requests
3) BAR/BOR/BUR vs. Time between updates

CHAPTER IV

FINDINGS

Three algorithms, maximum multicommodity flow (MMF), maximum concurrent flow (MCF) and general maximum flow (MAX) are simulated to see how each algorithm differs from the others under three situations: sensitivity to (1) the number of events, (2) average time between requests and (3) time between updates. The standard deviations from mean are from the simulations. The results are shown in the graphs (Figures 7 - 17). The error bars in some graphs cannot be seen because of their small values. The vertical lines shown in Figures 7 – 10, 12 – 15, and 17 are connected to the mean and are only given for visual confirmation.

4.1 Goodness Measures vs. Number of Events

An event is defined as a resource request. This simulation is to see how BAR, BUR and BOR change as the number of events increase. The number of events is incremented from 1000 to 40000 at intervals of 1000. The average time between requests is fixed as 0.05 seconds, update interval is every 500 seconds, and the average time a request keeps resources in the network is fixed at 10 seconds. The results shown are the average obtained after 50 runs.

4.1.1 BAR vs. Number of Events

The MAX and MMF draw almost the same lines (Figure 7). MCF follows a similar curve as MMF and MAX do, though the values are not even close. Starting with a gentle curve and staying flat for a moment, each algorithm decreases quickly and then keeps stable as the number of events increases. We can say that the BAR becomes lower and then stable as the number of events increase.

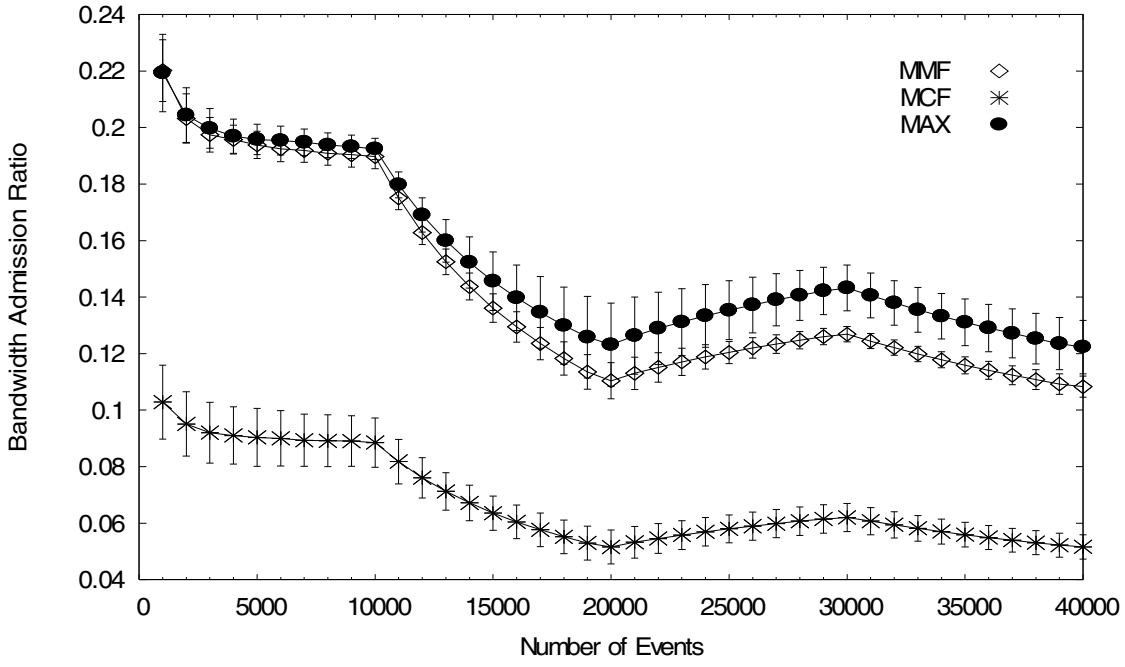


Figure 7. BAR vs. Number of Events

4.1.2 BUR vs. Number of Events

MCF stays high from the beginning compared to other algorithms (Figure 8). The BUR of MMF and MAX stays flat until some point and then increases. MMF keeps stable with a slight increase, and MAX falls over a narrow range of *y* as the number of events increase.

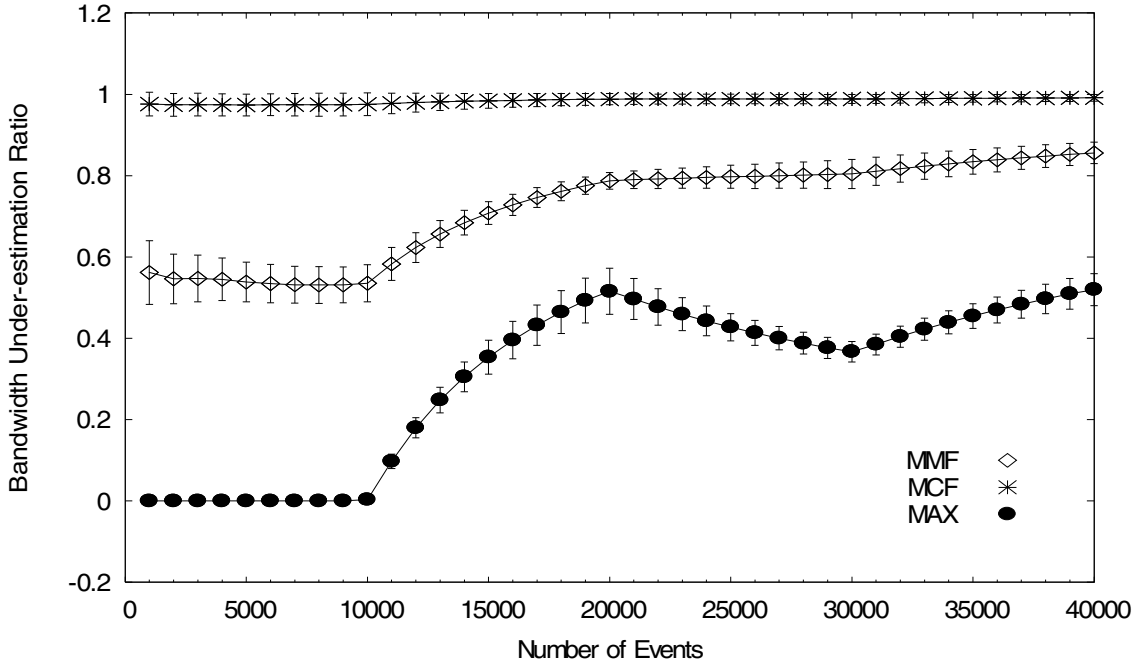


Figure 8. BUR vs. Number of Events

4.1.3 BOR vs. Number of Events

MAX stays high from the beginning to the end (Figure 9). On the other hand, the other two algorithms keep a low BOR. Though MMF decreases slightly, all three algorithms seem unaffected by the number of events simulated. The characteristics of this graph are that all lines stay flat, and the standard deviations from the simulations of all three algorithms are very small. MCF always has a value of zero. Therefore, the error bars cannot be seen.

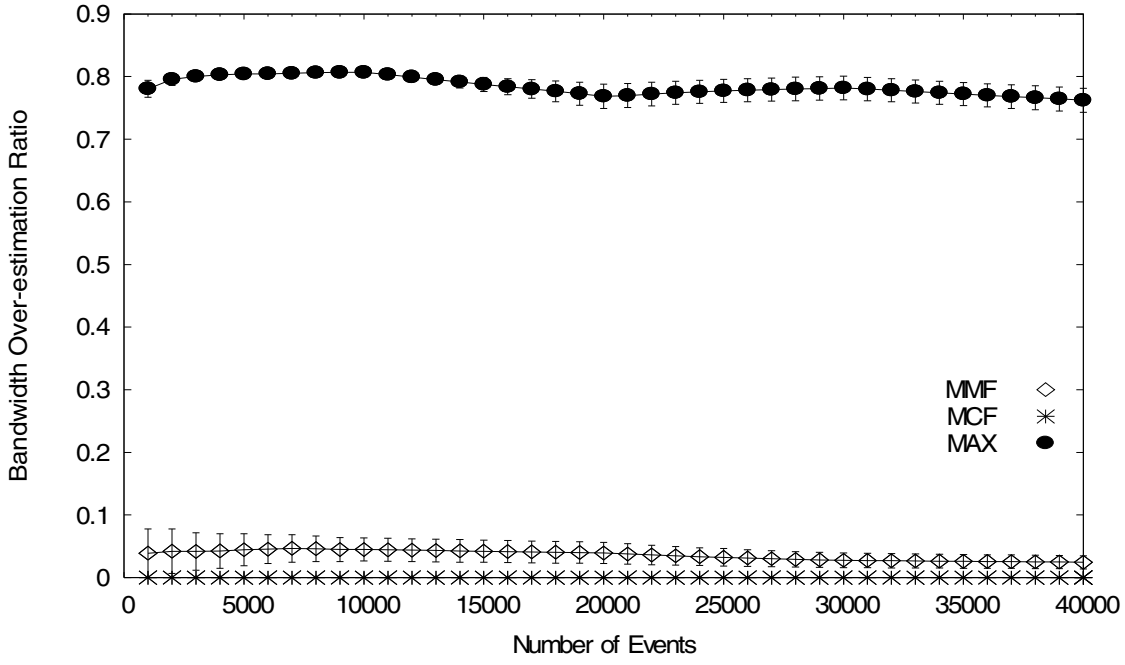


Figure 9. BOR vs. Number of Events

4.1.4 Discussion

As mentioned before, each edge's capacity is 4, the average time a resource is kept in use is 10 seconds, information is updated every 500 seconds, and the average time between requests is 0.05 seconds. We are able to estimate the time the simulation takes, and how many times network information is updated. The simulation times are calculated by multiplying the number of maximum event and the average time between requests, 0.05 seconds. At x = 1000, the simulation takes 50 seconds, 1000 times 0.05 equals to 50. In this case, network is updated just one time, at 0 second. Therefore, between x = 1000 and x = 9000, information is updated only one time, at 0 second. Between x = 10000 and x = 19000, information is updated two times, at 0 and 500 seconds. The network update affects performance of some algorithms at x = 10000, 20000, 30000 in BAR and BUR as graphs show, and we can see the changes of behaviors of goodness measures at these instances. BOR is not affected by update.

Though MMF and MAX show almost the same performance in BAR, the BOR of MMF is very low and that of MAX is very high. This tells us that MMF is more conservative than MAX.

MCF does not perform as well as the two others do in BAR. This is because of how it is implemented, and is explained in section 2.7. It makes advertised capacity low, and finally, ends up with poor performance.

4.2 Goodness Measures vs. Average Time between Requests

This scenario is to see how BAR, BUR and BOR change as the average time between requests increases. The results are recorded at x = 0.01, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45. The number of events, the time network information is updated, and the average time a request uses resources in the network are fixed at 20000 events, 500 seconds and 10 seconds respectively. The results under the standard deviations from mean due to simulations are the average obtained after 50 runs.

4.2.1 BAR vs. Average Time between Requests

The values of BAR of all three algorithms are almost the same at the beginning, but the differences increase as the average time between requests grows (Figure 10). The values of MAX are the highest, and those of MCF are the lowest. We can say that the less heavily loaded the network is, the better all three algorithms perform, especially MAX.

The unweighted nonlinear regression fits help to see the patterns (Figure 11). The data sets of MAX, MMF, and MCF are fitted well by $y = 1.185 * (1 - e^{-2.563x})$, $y = 0.6013 * (1 - e^{-4.156x})$, and $y = 0.4484 * (1 - e^{-2.528x})$ respectively. The equations tell us that MMF is approaching its asymptote faster than others; MAX and MCF increase at almost the same rate. Since the ratio cannot be above 1, the equation of MAX is unreasonable. The best fit with the coefficient constrained not to exceed 1.0 is

$y = 1.0 * (1 - e^{-3.369x})$, although the quality of this fit is not very good.

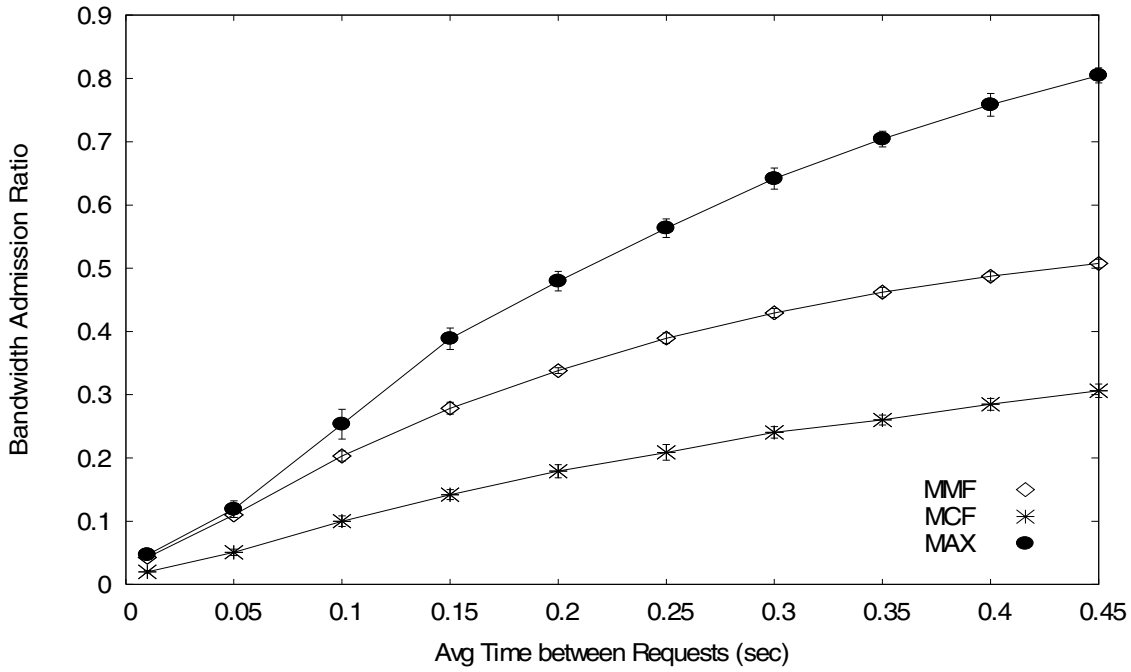


Figure 10. BAR vs. Average Time between Requests

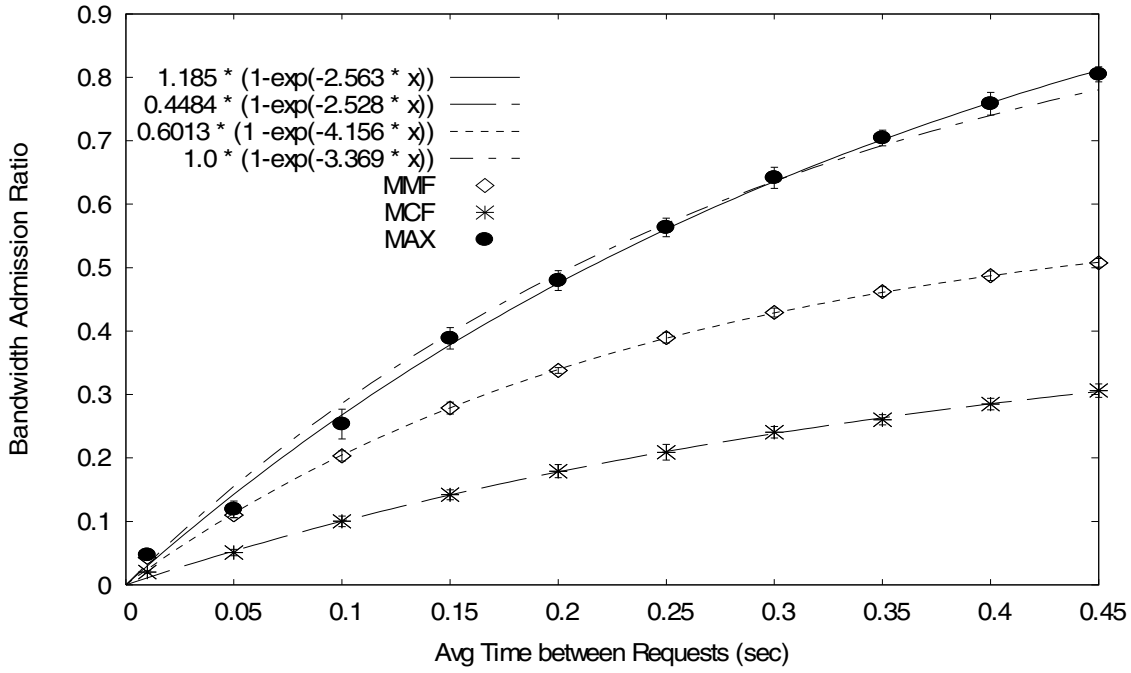


Figure 11. Unweighted Nonlinear Regression Fits

4.2.2 BUR vs. Average Time between Requests

MCF values do not change and keep a high ratio (Figure 12). The values of MMF start low with high standard deviations, get higher quickly, and keep their high values with small standard deviations. The values of MAX increase quickly after the beginning stage, and then seem to be stable, although they show some variation.

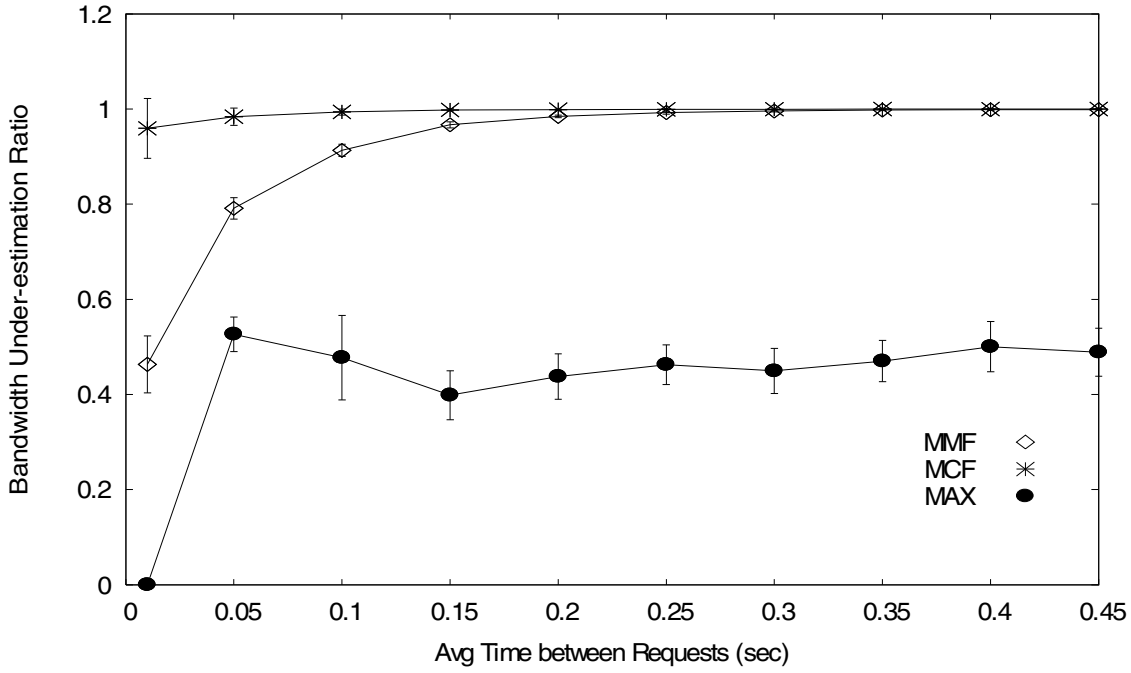


Figure 12. BUR vs. Average Time between Requests

4.2.3 BOR vs. Average Time between Requests

The values of MCF and MMF are always low (Figure 13). On the other hand, the MAX line shows dramatic changes; decreasing as the time increases. The standard deviations from mean for all three algorithms are extremely small, especially in longer average time between requests. This is why the error bars cannot be seen in the graph.

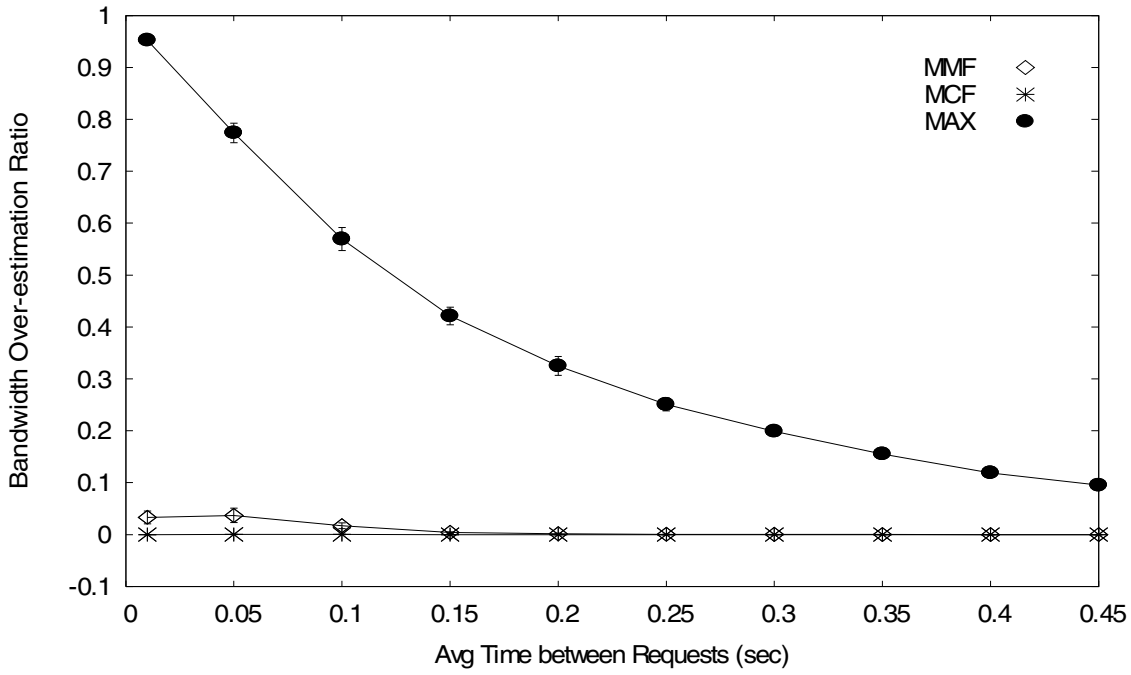


Figure 13. BOR vs. Average Time between Requests

4.2.4 Discussion

This simulation shows the influences of updates frequency as well as the network load.

We can estimate the time when the simulation ends by multiplying maximum number of event and average time between requests. In this simulation, 20000 is the number of maximum event. At $x = 0.05$, information is updated 2 times and 18 times at $x = 0.45$. Low x values means a heavily loaded network and high x values are for a lighter load.

In BAR, all algorithms result almost 0% at x = 0.01. Heavy load and infrequent updates cause this low ratio. Raw data shows us that numerator (accepted bandwidth) is extremely low. As the average time between requests increases, the network becomes less heavily loaded, and also the network information is updated more frequently. This reflects increases of the accepted bandwidth. As a result, BAR increases in all algorithms. The way each algorithm is implemented shows different increase rates in BAR graph. MAX is using general max-flow algorithm, so its returning advertised capacity may be large. As described in section 2.7, MCF will result in a low advertised capacity.

The business of network and frequency of updates do not influence BUR, but may have impacts on the BOR of MAX. As the network is less heavily loaded with more accurate network information, bandwidth dropped in the main network becomes less; therefore, we can see a declining line in Figure 13.

4.3 Goodness Measures vs. Time between Updates

This simulation is to see how BAR, BUR and BOR change as the time between updates increases. The results are recorded at x = 10, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500. The number of events is fixed at 20000 events, the average time between requests is 0.05 seconds, and the average time a request uses resource in the network is fixed at 10 seconds. The averaged values are obtained after 50 runs.

4.3.1 BAR vs. Time between Updates

The values of two algorithms, MMF and MAX, are especially unstable and their lines curve almost the same after x = 150 (Figure 14). The differences between mean of MMF and MAX are less than 2% after x = 200. As seen from the graph, the standard deviation of MAX is high after x = 250 and that of MMF is small. Therefore, the line of MMF and MAX could be much closer. The values of MCF seem to be stable.

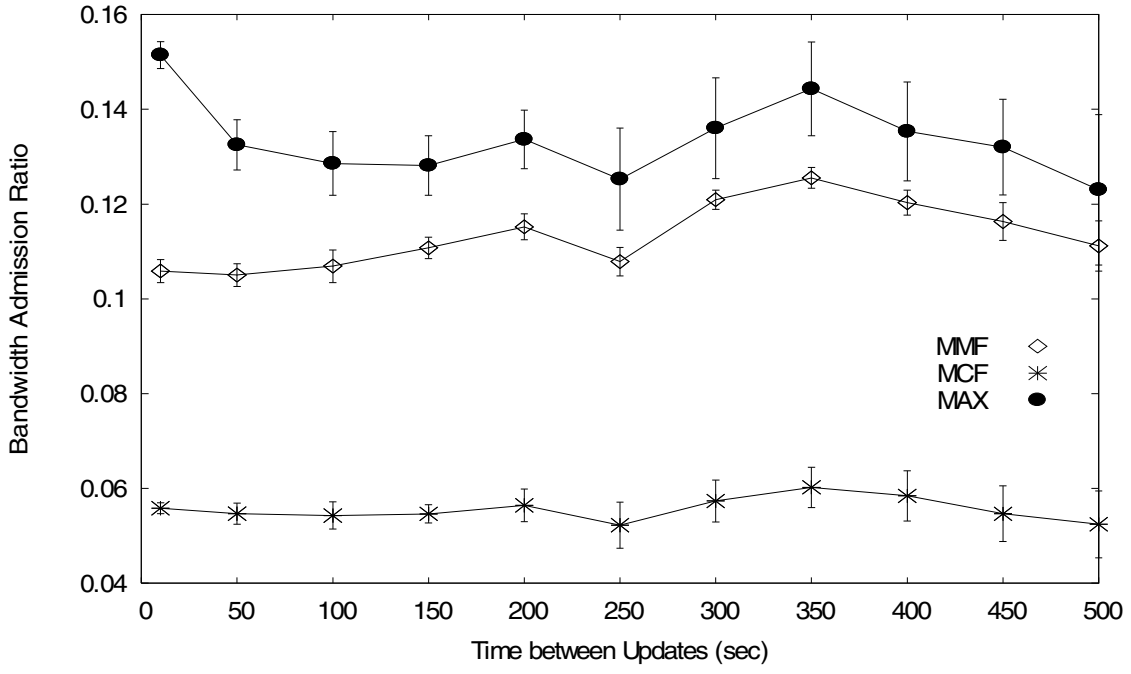


Figure 14. BAR vs. Time between Updates

4.3.2 BUR vs. Time between Updates

The values of MCF are the highest and MAX are the lowest (Figure 15). MMF has high values at the very short update intervals, but decrease gradually. The values of MCF keep flat. Those of MAX look unstable, but it seems they are within some range. BUR of MCF is inadequate, but that of MMF is not as bad as MCF. MAX displays reasonable result values.

Unweighted nonlinear regression fits show that the data set of MMF is fitted well by $y = 0.710 + (1 - 0.710) * e^{-0.00311x}$ in Figure 16. The equation says that y values approach to $y = 0.710$ as further simulation occurs.

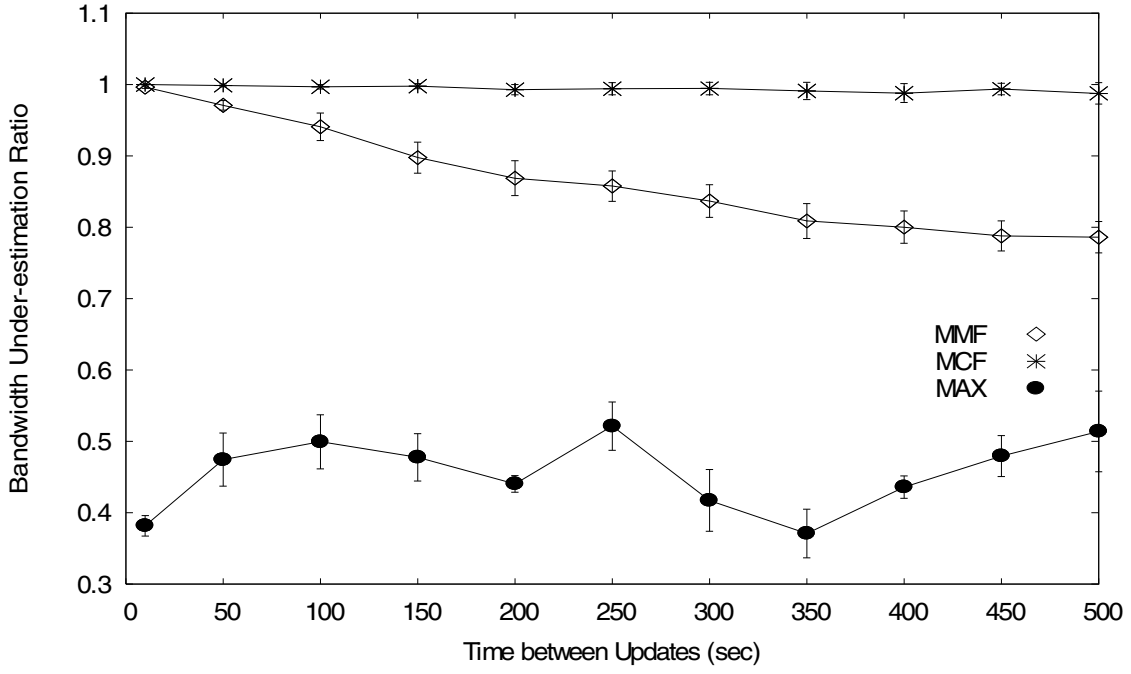


Figure 15. BUR vs. Time between Updates

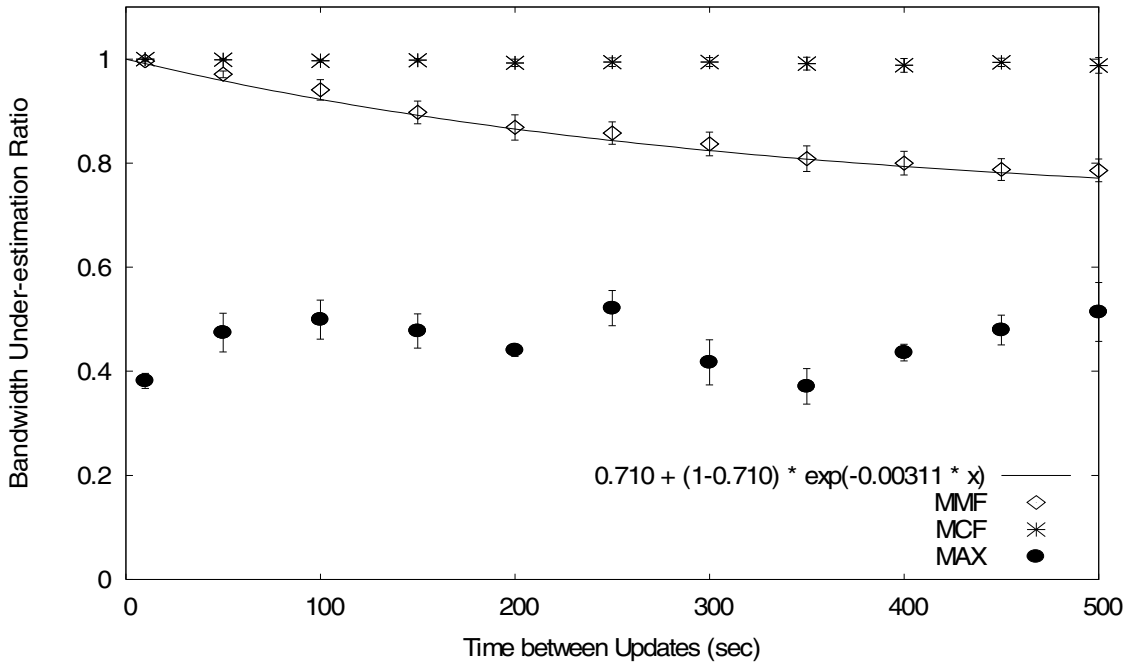


Figure 16. Unweighted Nonlinear Regression Fits with MMF

4.3.3 BOR vs. Time between Updates

The values of MAX are the highest among the three algorithms (Figure 17). The values of MMF gradually get higher and keep stable, and MCF stay extremely low from the beginning to the end. MAX begins with slightly low values, but the values increase gradually and keep flat.

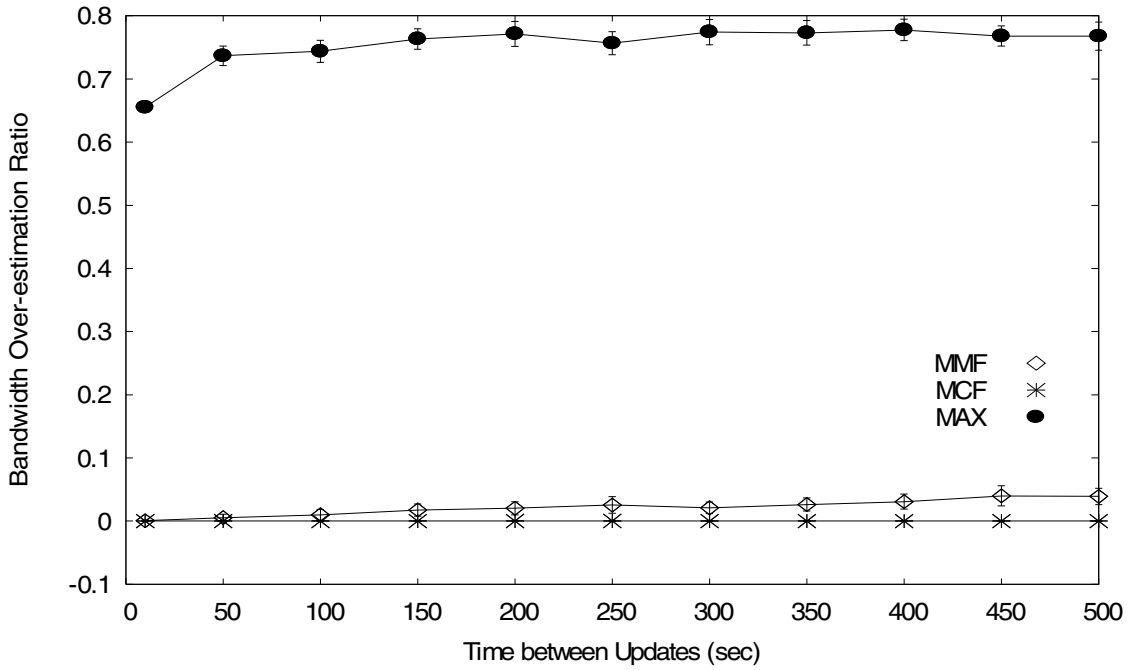


Figure 17. BOR vs. Time between Updates

4.3.4 Discussion

The simulation time is estimated by multiplying number of maximum event and the average time between requests; 20000 times 0.05 equal to 1000 sec. With this time and *x* value, we can see how many times network information is updated. The smaller the x value, the more updates occur. It seems that they do not affect the performance. There are no recognizable patterns on simulations except BOR. Here, also, how MCF is implemented seems to affect the performance.

4.4 Summary of Findings

We carried out simulations using three variables: (1) number of events, (2) time between requests, and (3) time between updates. Some graphs are obvious to see that they are affected by number of update times and network density. However, we cannot see recognizable patterns in some graphs.

In (1) and (3), MMF and MAX have almost the same performance in BAR. The BOR of MMF always shows an extremely low ratio. The way how each algorithm is implemented may help us to understand some of results. MAX calculates its advertised capacity by adding super sink, and returns a large number. On the other hand, MMF returns its advertised capacity so that each path can have maximum flow. Therefore, its number may not be as large as MAX's. This differences show well in some BOR graphs. We conclude that MMF is more conservative than MAX.

We can also explain poor MCF performance by how it is implemented. As mentioned in section 2.7, it returns a low advertised capacity. In conclusion, its performance is not as good as the other's in BAR. We are sure that a network topology with precise demands would increase its performance relative to the others.

CHAPTER V

CONCLUSION

The higher BAR and lower BOR and BUR are, the better the algorithm is. The motive of this research is to demonstrate that maximum multicommodity flow and maximum concurrent flow algorithms can result in lower BOR than the general maximum flow algorithm. This is because the general maximum flow algorithm finds a path from a single source to a single destination and does not allow sharing of resources which are requested from other domains. On the other hand, maximum multicommodity flow and maximum concurrent flow algorithms can deal with multiple sources and destinations and the problem of sharing resources with other domains. Their purpose is to maximize all paths' resource usage in the network. As we expected, maximum multicommodity flow and maximum concurrent flow algorithm reduced BOR. Also, we find that maximum multicommodity flow and general maximum flow algorithm produce almost the same results in BAR under heavy network usage with a long enough time between updates.

In addition to this, we can conclude that the maximum multicommodity flow algorithm is a more conservative approach than the general maximum flow algorithm. This is because the general maximum algorithm calculates advertised bandwidth capacities that can be forwarded *through* the main network; giving a tendency to return

large values. On the other hand, the advertised bandwidth capacities calculated by maximum multicommodity flow algorithm are reasonable in that it computes the values which can be routed with 100% guarantee to border routers in other domains.

As future work, various situations should be performed such as using a broad range of capacity of edges and request bandwidth, networks with more nodes and neighbor domains, dense networks, sparse networks, the length of time a resource is kept, etc. In addition to these, it would be exciting to simulate MCF again using demands of each commodity pair as an input.

Further development and analysis are needed for the QoS seeking algorithms to be used in the Internet [5]. I believe that this research can be another step toward the fast and efficient next generation network.

REFERENCES

[1] F. Hao and E. W. Zegura, "On Scalable QoS Routing: Performance Evaluation of Topology Aggregation," in *Proc. IEEE INFOCOM*, pp. 147-156, Mar. 2000.

[2] K.S. Lui, K. Nahrstedt, and S. Chen, "Hierarchical QoS routing in delay-bandwidth sensitive networks," in *Proc. of IEEE LCN*, Tampa, FL, pp. 176-189, 2000.

[3] L. K. Fleischer, "Approximating fractional multicommodity flow independent of the number of commodities," *SIAM J. Discrete Math*. vol. 13, no. 4, pp. 505-520, 2000.

[4] M. Bouklit, D. Coudert, J-F. Lalande, C. Paul, and H. Rivano, "Approximate Multicommodity Flow for WDM Networks Design," *SIROCCO '03*, 2003, pp. 43-56.

[5] M. Gopal, and K. Latha, "A QoS Engineering Architecture for the Next-Generation-Internet," *in Proc. of the International Conference on Computer Communications and Networks*, 1998, pp. 599.

[6] N.Garg and J. Könemann, "Faster and Simpler Algorithms for Multicommodity Flow and other Fractional Packing Problems," in *Proc. of the 39th Annual Symposium on Foundations of Computer Science (FOCS)*, Nov. 1998, pp. 300-309.

[7] S. Chen and K. Nahrstedt, "An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions," *IEEE Network*, vol. 12, no. 6, pp. 64-79, Nov.-Dec. 1998.

[8] T. H. Cormen, C. E. Leiserson, R. L. Riverst, and C. Stein, *Introduction to Algorithms, 2nd edition.* Cambridge, MA: MIT Press, 2001.

[9] V. Sarangan and R. Acharya, "A study on using network flows in hierarchical QoS routing," *GLOBECOM '01*, vol. 4, pp. 2188-2192, 2001.

[10] V. Sarangan, D. Ghosh, and R. Acharya, "Capacity-Aware State Aggregation for Interdomain QoS Routing," *IEEE Transactions on Multimedia*, vol. 8, no. 4, pp. 792-808, Aug. 2006.

[11] W. C. Lee, "Topology aggregation for hierarchical routing in ATM networks," in *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 2, pp. 82-92, Apr. 1995.

[12] W. Pak and S. Bahk, "Partial Optimization Method of Topology Aggregation for Hierarchical QoS Routing," *IEEE International Conference on Communications*, vol. 2, pp. 1123-1127, Apr. 2002.

[13] Y. Jia, I. Nikolaidis, and P. Gburzynski, "On the effectiveness of alternative paths in QoS routing: Research Articles," *International Journal of Communication Systems*, vol. 17, pp. 1-26, 2004.

VITA

Satoko Chika

Candidate for the Degree of

Master of Science

Thesis: USING MAXIMUM AND MAXIMUM CONCURRENT MULTICOMMODITY FLOWS FOR QOS ROUTING IN THE INTERNET

Major Field: Computer Science

Biographical:

Education: Received Bachelor of Science degree in Computer Science and Engineering from The University of Aizu, Aizu-Wakamatsu, Fukushima, Japan in March 2002; Completed the requirements for the degree of Master of Science in Computer Science at Oklahoma State University, Stillwater, July 2007.

Name: Satoko Chika Date of Degree: July, 2007

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: USING MAXIMUM AND MAXIMUM CONCURRENT MULTICOMMODITY FLOWS FOR QOS ROUTING IN THE INTERNET

Pages in Study: 39 Candidate for the Degree of Master of Science

Major Field: Computer Science

Once we connect the network cable to a plug-in jack on the wall, 'connecting' is a mundane event unless the service is unavailable. "How can we use limited resources efficiently in the Internet?" is an important question for future development. Efficiency is a requirement for next generation networks and will be a major benefit of QoS routing in the Internet. In this research, we simulated a simple network topology using general Maximum Flow, Maximum Multicommodity Flow and Maximum Concurrent Flow algorithms to calculate advertised bandwidth capacities. Under sensitivity to (1) number of events, (2) average time between requests and (3) time between updates, the results from the simulations showed that the Maximum Multicommodity Flow Algorithm is more conservative than a general Maximum Flow Algorithm. The Maximum Concurrent Flow Algorithm did not perform well compared to the others in this simulation.

ADVISER'S APPROVAL: Dr. Venkatesh Sarangan