End-to-End Delay in Localized QoS Routing

Ahmed S. Alzahrani
University of Bradford,
Dept of Computing, Bradford, UK
E-mail: a.s.m.alzahrani@bradford.ac.uk

Michael E. Woodward
University of Bradford,
Dept of Computing, Bradford, UK
E-mail: m.e.woodward@bradford.ac.uk

Abstract— Quality of service (QoS) routing has been proposed for supporting the requirements of network applications and satisfying connection constraints. A large amount of information needs to be exchanged periodically among routers. Therefore, in order to satisfy such requirements localized QoS routing algorithms have been proposed. This is where source nodes make routing decisions based on statistics collected locally. Using local information for routing avoids the overheads of exchanging global information with other nodes. In this paper we present a localized delay-based QoS routing (DBR) algorithm which relies on delay constraint that each path satisfies to make routing decisions. We also modify Credit Based Routing (CBR) so that this uses delay instead of bandwidth. The two localized algorithms and the commonly used global shortest path algorithm (Dijkstra's) are compared under different delay constraints and network topologies. We demonstrated through simulation that our scheme performs better than the modified CBR under different range of workloads and system parameters and outperforms the Dijkstra scheme in all network topologies, unless unrealistically small update intervals are used.

Keywords- Routing; Quality of Service; Localized Routing; Delay constraint.

I. INTRODUCTION

The concept of QoS routing has emerged from the fact that routers direct traffic from source to destination, depending on data types, network constraints and requirements to achieve network performance efficiency. It has been introduced to administer, monitor and improve the performance of networks. A lot of QoS algorithms [1] [2] [3] [4] are used to maximize network performance by balancing traffic distributed over multiple paths. Its major components include bandwidth, delay, jitter, cost, and loss probability in order to measure the end users' requirements, optimize network resource usage and balance traffic load.

The above QoS algorithms, which are source routing algorithms, depend on global network state information in order to make routing decisions. The global QoS network state needs to be exchanged periodically among routers. The efficiency of a routing algorithm depends on the accuracy of link state information [5], [6]. The balance between the frequency of update intervals and link state information is very important; the more update intervals of link state information the more accurate state information and vice versa. Such high levels of exchange may incur large communication and processing overheads.

However, packet switching networks provide good bandwidth sharing and data packets are transmitted with the least effort over the Internet. However, there is no guarantee for the application of the end-to-end delay, which is the sum of the delays at each hop along the path with intermediate nodes, and this may be required for real-time applications.

Localized QoS routing [7] [8] is proposed to achieve QoS guarantees and overcome the problem of using global network state information. Using such an approach, the source node makes its own routing decisions based on the information collected by monitoring the traffic generated from itself. Localized QoS routing does not need the global network state to be exchanged among network nodes because it infers the network state and avoids all the problems associated with it. In localized QoS routing each source node is required to first determine a set of candidate paths to each possible destination. It is not the intention of this paper to discuss this; however, more information about candidate path selection methods can be found in [9] and [10].

This paper proposes a delay based routing (DBR) which is a simple localized QoS routing algorithm that relies on average delay on the path in order to take routing decisions. We study other QoS routing schemes, such as the localized Credit Based Routing (CBR) proposed in [8] and [11]. We develop it using delay instead of bandwidth. We also use the global QoS routing scheme Shortest Path (Dijkstra) to compare their performance with our scheme in terms of flow blocking probability under different network loads and topologies.

II. RELEATED WORK

Although localized quality of service routing has been recently introduced as a new approach in the context of QoS routing, it is has previously only been studied using bandwidth metric in [7] and [8]. To the best of our knowledge, end-to-end delay has not been studied as the metric in localized quality of service routing algorithms. The two main localized quality of service routing algorithms that use bandwidth as the QoS metric are:

A. PSR

The Localized Proportional Sticky Routing (PSR) algorithm [7], which was the first localized QoS routing scheme. In this, each source node needs to maintain a set of candidate paths R. A path based on flow blocking probability and the load is proportionally distributed to the destination among the predefined paths. In PSR there are minimum hop (minhop) paths R^{min} and alternative paths R^{alt}, where $R = R^{min} \cup R^{alt}$. The PSR algorithm operates in two stages: proportional flow routing and computation of flow proportions. PSR proceeds in cycles of variable length, which form an observation period. Incoming flows are routed during each cycle along a path *r*, selected based on a flow proportion from a

set of eligible paths R^{alt}. Initially, all candidate paths are eligible paths and each of them is associated with an adjustable variable called the maximum permissible flow blocking parameter γ_r, which gives the maximum number of flows before the path becomes ineligible. For each minhop path, γ_r is set to *Y*, which is a configurable parameter, whereas alternative path γ_r is dynamically adjusted between 1 and *Y*. When all paths become ineligible, a cycle is completed and a new one is started after all parameters are reset. An eligible path is selected to route the flow based on its flow proportion. At the end of the observation period, a new flow proportion α_r is computed for each path in the candidate path set based on its observed blocking probability b_r. After each observation period the minhop paths flow proportions are adjusted to equalize their blocking probability ($\alpha_r b_r$). For the alternative paths, the minimum blocking probability among the minhop paths b* is used to control their flow proportion. That is, for each $r \in R^{alt}$, *if* $b_r < \Psi\ b^*$, $\gamma_r = \min(\gamma_r + 1, y)$. If $b_r > b^*$, $\gamma_r = max(\gamma_r - 1, 1)$, where Ψ is a configurable parameter to limit the 'knock-on' effect under system overloads. Note that $\gamma_r \geq 1$ ensures that some flows are routed along alternative paths to measure their quality [7] [8].

The PSR simulation results prove that it is simple, adaptive, stable, and it can be used as a good alternative to global QoS routing algorithms. However, there are difficulties for calculating flow proportions with non-Poisson traffic since the algorithm uses Erlang's loss formula to compute them, which relies on the steady-state distribution of blocking probability. Moreover, the number of candidate paths decreases in each cycle and the flow proportions need to be normalized when any path becomes ineligible. It is due to this that the flow distribution may not reflect the predetermined proportions, since the last paths that become ineligible tend to receive more flows than the others [7] [8].

B. CBR

The Credit Based Routing (CBR) [8] algorithm uses a simple routing procedure to route flows across the network. The CBR scheme uses a crediting scheme for each path in a candidate path set that rewards a path upon flow acceptance and penalizes it upon flow rejection. The path selection relies on the path's credits: the path with the larger credits among the candidate paths is chosen to send the flow. The CBR algorithm keeps updating each path's credits upon flow acceptance and rejection and does not compute a flow proportion. It also keeps monitoring the flow blocking probabilities for each path and adds this information in the crediting scheme to use in future path selection.

A set of candidate paths *R* between each source and destination is required in the CBR algorithm. Like PSR, CBR predetermines a minhop path set R^{min} and an alternative paths set R^{alt}, where $R = R^{min} \cup R^{alt}$. CBR selects the largest credit path *P*.credits in each set, minhop paths set R^{min} and alternative paths set R^{alt} upon flow arrival. The flow is routed along the minhop path that has the largest credit P^{min} which is larger than the alternative path that has the largest credit P^{alt}; otherwise the flow is routed along an alternative path using (1).

$$P^{min}.credits >= \Phi \times P^{alt}.credits \text{, where } \Phi \leq 1 \qquad (1)$$

Φ, is a system parameter that controls the usage of alternative paths. The CBR uses blocking probability in crediting schemes to improve the algorithm performance, as a path with low blocking probability will gain more credits. Path credits are increased and decreased upon flow acceptance and rejection respectively using blocking probability of the path. However, CBR uses a MAX_CREDITS parameter to determine the maximum attainable credits for each path using (2).

$$0 \leq \text{credits} \leq \text{MAX\_CREDITS} \qquad (2)$$

The CBR algorithm records rejection and acceptance for each path and uses a sliding window for a predetermined period of M connection requests. It uses 1 for flow acceptance and 0 for flow rejection, dividing the number of 0's by M to calculate each path blocking probability for M period.

III. THE PROPOSED ALGORITHM

This paper proposes two new localized routing algorithms. They are source routing algorithms, as the source nodes have the ability to select an explicit path to the destination node. The proposed algorithms assume that the network performs resource reservation and signaling message to set a path for new connections. In our proposed algorithms, end-to-end delay is used as the quality of service metric. The algorithms guarantee that the end-to-end delay which the flow experiences from its source node until the destination node never exceeds a certain constraint value.

A. CBR (delay)

The connection signaling in CBR starts when a new connection request arrives at the source node, which sends a setup message along the selected path. The message stores the delay over the outgoing link, and each intermediate node performs an admission test for the outgoing link and adds the outgoing link delay to the previous delay. If the delay that the message experiences is less than the QoS delay, the delay is reserved for that flow and the message is forwarded to the next node. The flow is accepted if the delay experienced in the selected path is less than the QoS delay, as in (3), which means that end-to-end over that path satisfies the delay constraint. Otherwise, if the delay over the selected path exceeds the QoS delay on any part of the path, a failure message is propagated back to the source node and the flow is rejected. This means that the delay over that path does not satisfy the delay constraint.

$$\sum_{i \in n} delay\ (i) \leq QoS\ \_\ Delay \qquad (3)$$

where n is the number of links along the selected path.

The information regarding flow acceptance or rejection is acknowledged to the source node in order to collect flow statistics. The pseudo code for CBR algorithm is as follows:

```
PROCEDURE CBR ( )
  Initialize
  Set P.credits = MAX_CREDITS, ∀  P ∈ R
```

CBR ( )

1. If P.credits =0 $\forall$ P $\in$ R
2. Set P.credits = MAX_CREDITS, $\forall$ P $\in$ R
3. P^{min} =max {P.credits: $P \in R^{min}$ }
4. P^{alt} =max {P.credits: $P \in R^{alt}$ }
5. if P^{min} .credits >= $\Phi \times P^{alt}$ .credits
6. set $P = P^{min}$
7. else
8. set $P = P^{alt}$
9. Route flow along path P
10. if sum {L.delay : L $\in P$} $\leq$ QoS_Delay
11. UpdateBlockingProbability(P)
12. amount =(1- P.BlockingProbability(P)
13. P.credits=min{ P.credits + amount, MAX_CREDITS}
14. else
15. UpdateBlockingProbability(P)
16. amount =(1- P.BlockingProbability(P)
17. P.credits=min{ P.credits - amount, 0}

END PROCEDURE

Like most localized QoS routing algorithms, CBR requires every node to maintain a predetermined set of candidate paths R to each possible destination. CBR distinguishes between two types of paths: the set of minhop paths R^{min} and the set of alternative paths R^{alt}, where $R = R^{min} \cup R^{alt}$. P.credits is a variable associated with each candidate path. $P \in R$ stores credits for each candidate path. P.credits is set to MAX_CREDITS, which is a system parameter that determines the maximum credit each candidate can gain. CBR selects two paths upon flow arrival from each set of candidate path, P^{min} (line 3) and P^{alt} (line 4), which are the paths with maximum credits in the minimum candidate paths set R^{min} and the path with maximum credits in the alternative candidate paths set R^{alt}.

The flow is routed along the path with the largest credit P^{min} among minimum candidate paths set, if P^{min} .credits >= $\Phi \times P^{alt}$ .credits (lines 5-6); where $\Phi \leqslant 1$, otherwise, P^{alt} is chosen (line 8). Φ is a system parameter that controls the usage of alternative paths and limits the 'knock-on' effect. The flow is accepted if the end-to-end delay along the selected path satisfies the QoS delay (delay constraint) (line 10), as described earlier in (3). The blocking probability for the selected path *P.credits is* updated by increasing its credit with an amount that corresponds to its success probability (line 11-13) when the flow is accepted. Whereas if the flow is rejected, the blocking probability for the selected path *P.credits is* updated by decreasing its credit with an amount that corresponds to its failure probability (line 15-17)

CBR uses a sliding window to calculate the blocking probability with a predetermined size, *W (*connection requests). It keeps monitoring the flow blocking probabilities and records them upon flow acceptance and flow rejection. So, for each path of the candidate paths set, the blocking probability is computed for the recent period. The sliding window *W* moves to get the recent value by inserting it at the head of the list and removing the oldest one from the rear of the list. As an example, let us suppose the list of recent five acceptances and rejections is {1, 0, 1, 0, 0}, where 1 represents flow acceptance and 0 represents flow rejection. In this case the blocking probability is 3/5, and the oldest and newest values are 1 and 0 respectively. Let us also suppose that the new flow is rejected; blocking probability will then be 4/5 as the newest rejected flow, 0, will be inserted and the oldest value, which is 1, will be removed from the list. The new updated list will be {0, 1, 0, 0, 0} and the blocking probability will be updated accordingly.

Although CBR shows good performance against other localized QoS routing PSR using bandwidth as the QoS metric [14] and in terms of using the delay QoS metric (as we will see in the evaluation of our CBR (delay) algorithm), the criteria used for path selection in CBR, which relies on a crediting scheme, does not directly reflect on the quality of a path. Logically, the quality of a path should be measured directly based on the required QoS metric, like bandwidth or delay. This is why the second localized algorithm is proposed.

B. DBR

The second new localized QoS routing algorithm proposed in this paper is the Delay-Based Routing (DBR) scheme, which relies on the average delay in the path in order to take routing decisions. Unlike PSR and CBR, whether in bandwidth metric or delay metric (which performs routing decisions based on flow statistics of path blocking probability), DBR uses a completely different scheme in terms of path selection based on collection of statistics about the actual delay in each candidate path. Calculating the average delay for each candidate path is then used to measure the quality of the path, and upon flow arrival the path with the least average delay is used to route the incoming flow. DBR keeps monitoring the delay in the network and continuously updates each path's average delay in the candidate path set. Using the average delay for the path directly reflects on the actual quality of the path.

When a new connection arrives at a source node the signaling process starts to select a path to route a flow. The source node computes the path that may satisfy the QoS delay requirements. It uses a setup message to travel along the selected path with each connection request. The message stores the delay over the outgoing link, and each intermediate node performs an admission test for the outgoing link, adding the outgoing link delay to the previous delay. If the delay that the message experiences is less than the QoS delay, the delay is reserved for that flow and the message is forwarded to the next node. The flow is accepted if the delay experienced in the selected path is less than the QoS delay, as in (3), which means that end-to-end over that path satisfies the delay constraint. Otherwise, if the delay over the selected path exceeds the QoS delay on any part of the path, a failure message is propagated back to the source node and the flow is rejected. This means that the delay over that path does not satisfy the delay constraint. The information regarding flow acceptance or rejection is acknowledged to the source node in order to collect

flow statistics. The pseudo code for DBR algorithm is as follows:

```
PROCEDURE DBR ( )
  Initialize
    Set P.average = 0, ∀   P ∈  R
  DBR ( )
    1.    set P=min {P.avg: P ∈  R}
    2.    Route flow along path P
    3.    if  sum {L.delay  : L ∈ P} ≤ QoS_Delay
    4.        Calculate Average delay (P)
    5.        P.delay=sum {L.delay  : L ∈ P}
    6.        Value=(value + P.delay)
    7.        P.avg= {P.avg, Value}
    8.    else
    9.        Calculate Average delay (P)
   10.        P.delay=sum {L.delay  : L ∈ P}
   11.        Value=(value + P.delay)
   12.        P.avg= {P.avg, Value}
END PROCEDURE
```

Each source-destination pair in localized delay-based QoS routing requires a predefined set of candidate paths ***R***. The main characteristic that is associated with every path ***P*** in the candidate path set is the average delay. We use P.avg to store the average end-to-end delays and update its value with every connection request. Upon flow arrival, DBR selects the path with the smallest average delay (line 1) and routes the flow along the selected path. As the setup message travels to the destination it adds the outgoing link for each hop that the message passes. At the same time it performs a comparison over the links along the path with quality of service delay to ensure that this path satisfies the delay constraint (lines 2-3). If the flow is accepted along the selected path, the end-to-end delay is calculated along that path, and the path delay is then added to the previous (delay) values of the path and stored in the source node (lines 4-7). As a new connection arrives to the source node, the stored values are divided by the number of connections sent in order to get the actual delay of the path. It should be noted that storing the end-to-end delay along the selected path reflects on the actual delay that the path can support. In contrast, if the flow is rejected the delay calculated so far, which is larger than the delay constraint, is added to the overall path delay and the new average is calculated as previously (lines 9-12). So, when the path's delay is increased its probability to be chosen is decreased for new connections. Increasing or decreasing the path delay reflects on the actual path state and the quality of the path can be measured accurately.

Unlike CBR, which monitors flow blocking probabilities, DBR monitors delay of a path and the source node stores delay values for the accepted or rejected flow of each path. It calculates the average delay using a simple moving average (sliding window) over a predetermined period. DBR uses a fixed size of the sliding window of size w connection requests, which moves to get the most recent value by inserting it at the head of the list and removing the oldest one from the rear of the list. So, for the sliding window, the average delay will be calculated using the most recent *W* connection requests. For example, if {10, 15, 8, 13, 9}, represent the last five delays collected over a period W=5 for path P, the average delay that the path P could support would be (10+ 15+8+13+ 9)/5=11, and the oldest and newest values are 10 and 9 respectively. Let us also suppose that the new arrival is rejected and the end-to-end delay of the path was 16. The set will be changed to {15, 8, 13, 9, 16} and the new average delay will be (15+8+13+ 9+16)/5=12.2.

IV. PERFORMANCE EVALUATION

This section evaluates the performance of the proposed localized algorithms: the credit-based routing scheme (CBR) and delay-based routing scheme (DBR). A global routing algorithm (Dijkstra) was also used in the comparison, since many contemporary routing algorithms, such as OSPF, are based on this. Dijkstra's algorithm finds the shortest path in terms of delay that satisfies the quality of service delay constraint (we use the notation Dijkstra(x) to refer to this algorithm with update intervals of x time units for link state information and Dijkstra to refer to the optimal result).

A. Simulation model

We implemented our localized QoS routing schemes (DBR and CBR) based on the discrete-event simulator OMNeT++ [12], and conducted extensive simulations to test their performance. Using one of the predetermined algorithms (DBR, CBR, and Dijkstra's), the simulation performs path selection, resource reservation, and admission control at flow level.

Due to the varying performance of algorithms with underlying network topologies, we have also used different types of network topologies. We used an ISP topology in the simulation, which is widely used in different QoS routing algorithm studies [13] [14]. A 49-node torus and lattice regular topologies were also used in the simulation to be able to select different path lengths between each source-destination pair [14] [15]. Random topologies were created using C++ and OMNeT++, where the connection between any two nodes is determined by a probability using the Doar-Leslie Model [16]. This model is based on the Waxman Model using a scaling factor, which stabilizes the node degree of the graph. Table 1 lists the most important characteristics of the topologies used in the experiments.

Topology	Nodes	Links	Node degree	Avg. path length
RANDOM80	80	484	6.07	2.9212
RANDOM60	60	326	5.43333	2.568
ISP	32	108	3.375	3.177
Torus	49	196	4	3.5
Lattice	49	168	3.42857	4.66667

Table 1: Network topologies and their characteristics

B. *Traffic generation*

In each experiment in the simulation the network topology remains fixed. The traffic is generated by having the source node choose the destination node from amongst all nodes, except the source node, with uniform probability. Flow interarrival times at the source node are exponentially distributed with the mean $1/\lambda$. The mean of the flow holding time is $1/\mu$, with an exponential distribution, while the QoS delay is varied, ranging between 5 and 30 time units. We also assume that all links in the topologies are bidirectional and the mean delay time for each link is also exponentially distributed.

The parameters used in the simulation for CBR are MAX_CREDITS=5 and Φ=1. Blocking probabilities are calculated based on the most recent 20 flows for DBR and CBR. Candidate paths between each source-destination pair in a network topology are chosen, so we include minimum hop and (minimum hop) +1, +2 in the set to get the required number of candidate paths between each pair. All experiments collected result from at least 2,000,000 connection requests (arrivals).

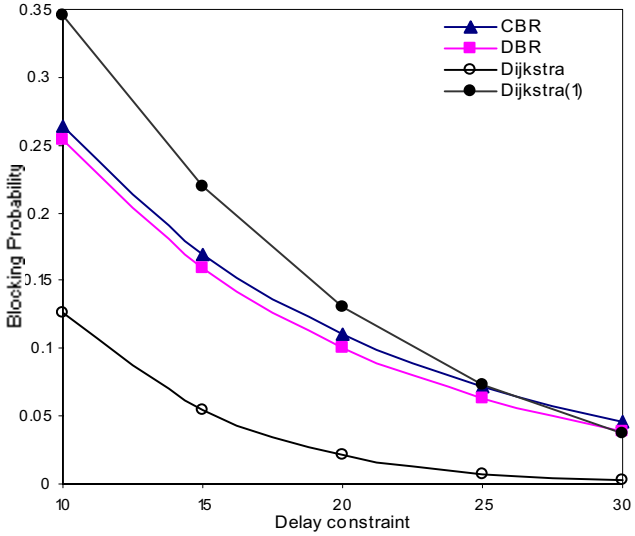
C. *Performance metrics*

Flow blocking probability was used to measure the performance of the algorithms. Flows will be rejected when the path does not satisfy the delay constraint. The blocking probability is defined as:

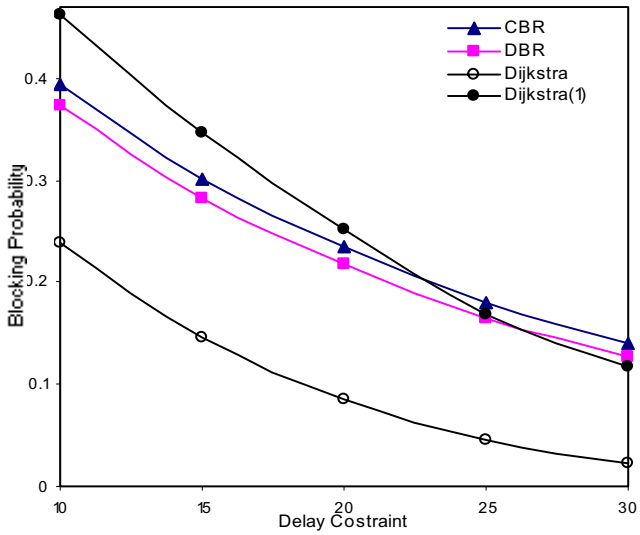
$$\text{Flow blocking probability} = \frac{\text{No of rejected requests}}{\text{No of requests arriving}}$$

D. *Simulation results*

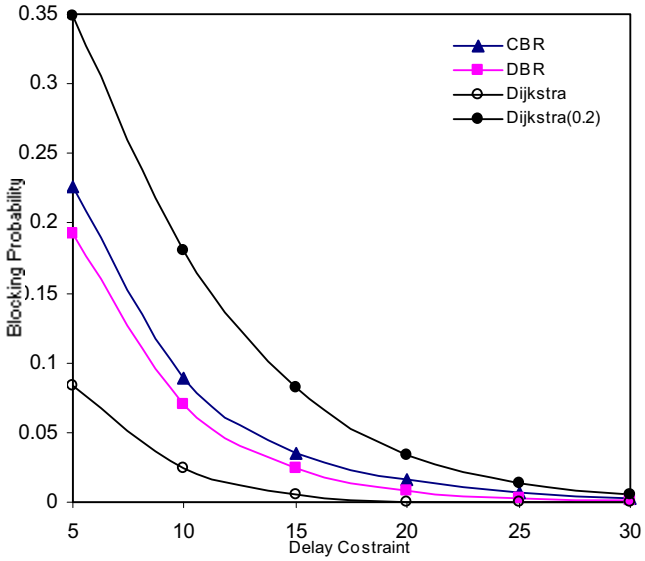
i) *Blocking probabilities*



(a) ISP



(b) Lattice



(c) Random 60

Fig. 1. Delay constraint in different network topologies.

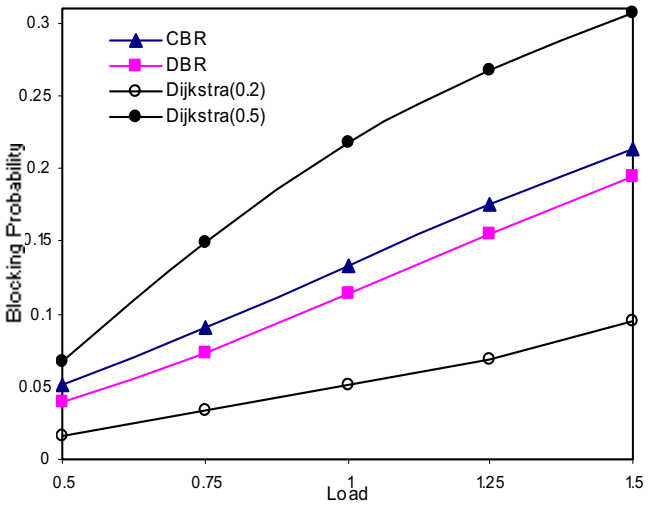
Fig. 1. Shows flow blocking probability plotted against different ranges of delay constraints for different types of network topologies. The characteristics of these topologies were described earlier in table 1. It can be noted that all algorithms satisfy most flows under large delay constraint (constraint 30), which can be expected as the probability of finding a path that satisfies a large delay constraint is high and most flows will be accepted. However, performance under some constraints may be significantly degraded based on the average path length in the topology. This can be noticed in fig. 1. (b), with Lattice topology giving poor performance compared with the other topology under a delay constraint of 30 due to the fact that it has the largest average path length. Whereas in fig. 1. (c), all algorithms give superb performance with the same constraint as Random 60, which has the smallest average path length. The blocking probability increases gradually as the constraint tightens over incoming flows, which

implies that flows under small delay requirements are hard to route, as expected.

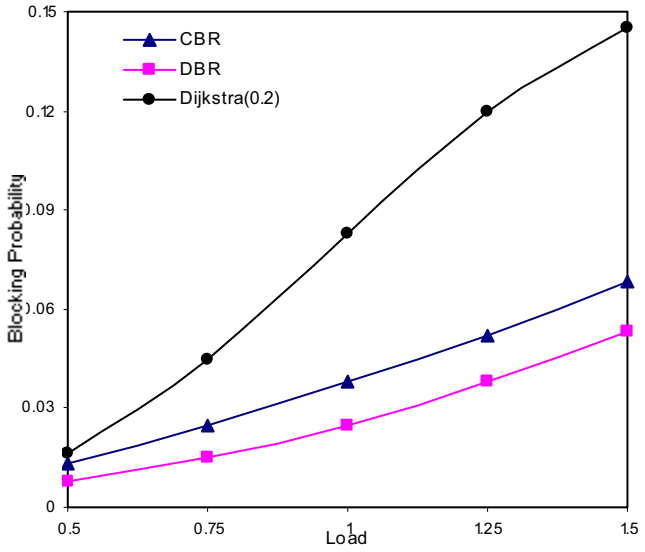
The performance of Dijkstra's algorithm is significantly affected by the update interval. We can see that as the update interval of the global state information increases, its performance degrades significantly and its blocking probability increases rapidly. This is due to the path selection of Dijkstra's algorithm, which is based on the periodic update of QoS global state information that does not respond quickly to the change in network state and sticks with the current feasible path until the next update interval becomes available.

In the case of localized routing algorithms CBR and DBR, we can notice that both algorithms perform well in all network topologies. Their blocking probabilities increase gradually as the delay constraint increases; which is not the case in Dijkstra's algorithm, where they increase sharply under the same constraint. This is due to the effect of alternative routing, which does not rely on global state information for path selection as in Dijkstra's algorithm. CBR selects the path with the maximum credits as long as it does not reject flows, since credits of the selected path are changeable according to blocking probability. This leads the CBR to select alternative paths with the updated credit. However, rejection flow will cause the choosing of an alternative path with more credits. In the same way, DBR selects paths with the least average end-to-end delay, which gives more scope to select paths as long as they satisfy QoS delay. On the other hand, the DBR mechanism avoids the crediting scheme associated with the CBR scheme by selecting the path based on its quality satisfying QoS delay. The path selection method used in DBR performs well under varying delay constraints and network topologies when compared to CBR and Dijkstra's algorithm with small update intervals, as shown in fig. 1.

ii) Impact of network topologies and varying arrival rates



(a) Torus



(b) Random 60

Fig. 2. *Impact of varying arrival rates in network topologies*

In fig. 2 the flow blocking probability is plotted against different ranges of arrival rate for different types of network topologies. We start by offering a small arrival rate, 0.5, whereby all algorithms can accept most of the arrivals as long as the average path length is small. We then increase the rate to see how the local and global algorithms will perform. From this figure we can notice that in Random 60, CBR and DBR schemes give superb performance compared with Dijkstra's algorithm, even with a very small update interval of global state information (0.2). However, because of path selection in the DBR algorithm, which relies on the end-to-end delay on a path, its blocking probability is better than CBR again in all topologies. When the arrival rate increases, CBR and DBR adapt to the change and maintain their relative performance. This can be seen in random topologies, as the blocking probability increases gradually, whereas Dijkstra's algorithm can't react promptly to changes in arrival rates and performs poorly as the arrival rate increases. This can be expected as the periodic updates do not respond quickly enough to rapid variations in flow arrival. In the case of the torus topology, DBR and CBR fail to perform better than Dijkstra (0.2), which is a very small update interval of link state, but still give good results against larger update interval (0.5). This is most likely because the Torus is a regular topology and there would be less likelihood of route flapping than with Dijkstra's algorithm.

iii) Impact of heterogeneous delay constraint

The experiments so far have seen only one delay constraint being used as QoS delay. In the following we study the affect of using more than one delay constraint on Dijkstra, CBR, and DBR. We consider two delay constraints in order to study the impact of large and small constraint flows, but having the same inter arrival rate and holding time. The value of the delay constraints for both types is 20 for the large constraints and 10 for the small constraints, with a mean inter-arrival rate of 1.

The holding times for all flows are exponentially distributed with a mean of 2.5. Performance is measured by mixing fractions of small and large delay constraints, keeping the inter arrival mean fixed at 1.

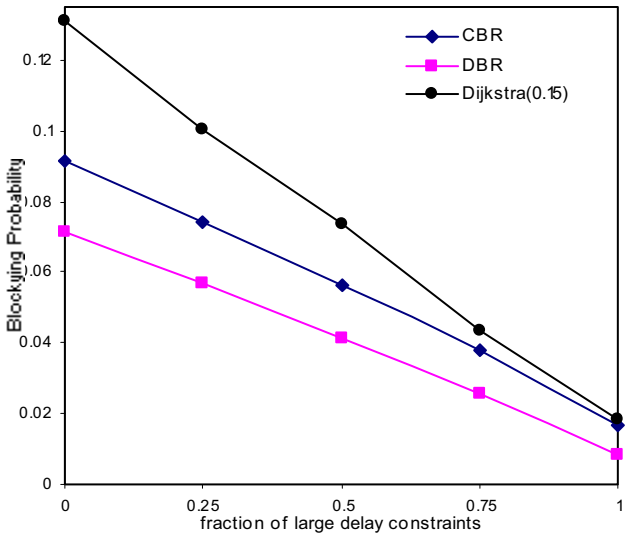


Fig. 3. *Impact of heterogeneous delay constraint in* Random 80

Fig. 3 shows flow blocking probability plotted against the fraction of large delay constraints. It can be noticed that the blocking probability of the three algorithms is decreased as the fraction of large delay increases in random 80, which is expected since it is easier for a source node to find a feasible path with a large delay constraint. Dijkstra gives poor performance, regardless of the fraction of small delay, unlike DBR which gives the best performance. This can be expected, as DBR continuously monitors the end-to-end delay in each path and the requested QoS delay is known before path selection for both large and small delay constraints. CBR lies in the middle and selects the path with the maximum credits, as long as it does not reject flows, since credits of the selected path are updated after every flow routed along the path. However, any flow rejection will cause its credits to be decreased and an alternative path with more credits to be selected. It can also be noticed that the difference in performance remains fixed between CBR and DBR, and they have good performance. Dijkstra's algorithm for Random 80 has 0.15 update intervals, which is a very small update interval.

iv) DBR and CBR time complexity

Global QoS routing algorithms performing a variant of Dijkstra's algorithm to find the shortest path or widest path, or both like WSP, take at least O (N log N+L) time; where N is the number of nodes and L is the number of links in the network. On the other hand, the complexity of selecting a path among the set of candidate paths R in CBR and DBR is O (|R|). CBR needs to update blocking probabilities, which takes a constant time O (1). Similarly, DBR needs to update the average delay, which also takes O (1).

V. CONCLUSION

The localized QoS routing scheme has been proposed to overcome the problems associated with global QoS routing. In localized routing, decisions are made based on the local view of the network QoS state. We have developed two localized QoS routing algorithms: CBR, which uses a simple crediting scheme that is increased upon flow acceptance and decreased upon flow rejection; and DBR, which relies on actual delay on the path in order to take routing decisions. We demonstrated through simulation that the two proposed algorithms, although simple, outperform global routing schemes under different traffic loads and network topologies, even for a small update interval of link state. Our general results suggest that localized QoS routing should be based on schemes that explicitly reflect the quality of a path, rather than schemes that are based indirectly on the path quality. Moreover, localized QoS routing can be employed to routes in a network with multi-constraint delay requirements or heterogeneous traffic.

REFERENCES

[1] G. Apostolopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda, and T. Przygienda., "QoS Routing Mechanism and OSPF Extensions", in RFC 2676, 1999.

[2] Q. Ma, P. Steenkiste, "Quality-of-Service Routing for Traffic with Performance Guarantees", Proceedings of the IFIP Fifth International Workshop on Quality of Service, New York, 1997.

[3] Z. Wang and J. Crowcroft, "Quality-of-Service routing for supporting multimedia applications", IEEE J. Select. Areas Commun, vol. 14, pp. 1228-1234, 1996.

[4] Q. Ma, P. Steenkiste, and H. Zhang, "Routing High-bandwidth Traffic in Max-min Fair Share Networks", Proceedings of ACM SIGCOMM'96, Palo Alto, CA, 1996.

[5] G. Apostolopoulos, R. Guerin, S. Kamat, and S.Tripathi, "Quality of Service Based Routing: A Performance Perspective," Computer Communication Review, vol. 28, 1998.

[6] R. Guerin and A. Orda, "QoS-Based Routing in Networks with Inaccurate Information: Theory and Algorithms," IEEE/ACM Transactions on Networkinlg, vol. 7, pp. 605-614, 1999.

[7] S. Nelakuditi, Z. L. Zhang, R. Tsang, and D. Du, "Adaptive Proportional Routing: a Localized QoS Routing Approach", IEEE/ACM Transactions on Networking, vol. 10, pp. 790--804, 2002.

[8] S. Alabbad, and M. E. Woodward, "Localized Credit Based QoS Routing", Proceedings of IEE, vol. 153, PP. 787-796, 2006.

[9] X. Yuan, and A. Saifee, "Path Selection Methods for Localized Quality of Service Routing", the 10th IEEE International Conference on Computer Communications and Networks (IC3N 2001), Phoenix, Arizona, 2001.

[10] S. Nelakuditi, Z.L. Zhang, R. Tsang, and D. Du, "On selection of candidate paths for proportional routing", Computer Networks, vol. 44, pp. 79-102 2004.

[11] S. Alabbad, and M. E. Woodward, "Localized Credit Based QoS Routing: Performance Evaluation Using Simulation", Proceedings of the 40th Annual Simulation Symposium, IEEE Huntsville, AL April 2-6, 2006.

[12] A. Varga, "OMNeT++ Discrete Event Simulation System Version 3.1. User Manual", in URL: http://www.omnetpp.org/doc/manual/usman.html, 2007.

[13] G. Apostopoulos, R. Guerin, S. Kamat, A. Orda, and S. K. Tripathi, "Intradomain QOS routing in IP networks: A feasibility and cost/benefit analysis", vol. 13, pp. 42–54, Sept.-Oct 1999.

[14] S. Chen and K. Nahrstedt. On Finding Multi-constrained Paths. In Proceedings of ICC'98, pages 874–879, June, 1998.

[15] F. A. Kuipers et al., "Performance Evaluation of Constraint-Based Path Selection Algorithms", IEEE Network, September/October, 2004.

[16] M. Doar and I. Leslie, "How Bad is Naïve Multicast Routing?" IEEE INFOCOM 1993, pp. 82-89, 1993.