Localized Credit Based QoS Routing : Performance Evaluation Using Simulation

Saad H. Alabbad, M.E Woodward
Department of Computing, School of Informatics
University of Bradford, Bradford, BD7 1DP, UK
{S.Alabbad, M.E.Woodward}@Bradford.ac.uk

Abstract

Localized Quality of Service (QoS) routing has recently been proposed as a viable alternative approach to traditional QoS routing algorithms that use global state information. In this approach, problems associated with maintaining global state information and the staleness of such information are avoided by having the source nodes to infer the network QoS state based on flow blocking statistics collected locally, and perform flow routing using this localized view of the network QoS state . In this paper we introduce a credit based routing algorithm (cbr) which is a simple yet effective localized QoS routing algorithm. We compare its performance against the localized proportional sticky routing (psr) algorithm using different types of network topologies, QoS requirements and traffic patterns and under a wide range of traffic loads. Extensive simulations show that our algorithm outperforms the psr algorithm with the same time complexity.

1. Introduction

The two dominant approaches for QoS routing are: source routing and distributed routing. In most source routing algorithms [4, 21, 10, 8, 14, 17, 1], each source node must have global QoS state information of the network in order to perform routing. This global state is typically updated periodically by a link-state algorithm and maintaining it up-to-date gives raise to several problems. First, state information at every source node has to be updated frequently enough to reflect the flow dynamics, which imposes a large communication overhead and significantly affects scalability and the performance of this approach. Second, due to this overhead and non-negligible propagation delay of state messages, the link-state algorithm can only provide an approximate global state. This inaccuracy has a significant impact on the performance of routing algorithms [18] and needs to be considered when designing routing algorithms [8]. Third, it was shown in [18] that out-of-date information due to large update intervals can cause route flapping in global QoS routing algorithms. That is, when the utilization on a link is low, an update causes all the source nodes to prefer routes along this path, resulting in a rapid increase in its utilization. Similarly when the utilization is high, an update causes all the sources to refrain from using this link and consequently its utilization decreases rapidly. Forth, the computational overhead at the source is very high considering that QoS routing is typically done on a per-flow basis.

Distributed algorithms which need global state share the aforementioned drawbacks of source routing, while algorithms that do not require QoS global state usually flood the network with excessively large numbers of control messages to compensate for lack of global state [9, 2, 3, 19] and the scalability of such algorithms is limited by the prohibitive message overhead.

Localized QoS routing [13] is a recently proposed approach that tries to circumvent these problems by having the source nodes to infer the network QoS state based on flow statistics collected locally, and perform flow routing using this localized view of the network QoS state. In localized routing, each node maintains a predetermined set of candidate paths to each possible destination and routes flow along these paths. The selection of the candidate paths is a key issue in localized routing and has a considerable impact on how the localized routing algorithm performs. Various methods for the selection process and their performance evaluation can be found in [22, 12].

In this paper we introduce a credit based routing algorithm (cbr) which is a simple yet effective localized QoS routing algorithm. We compare its performance against the proportional sticky routing (psr) algorithm proposed in [13] and the widest-shortest path algorithm [1] which is a global QoS routing algorithm. We study their performance using different types of topology (real, random and regular topologies), traffic patterns (Poisson and bursty traffics) and under a wide rang of traffic loads. All routing algorithms presented in this paper use bandwidth as the only metric for routing since it is one of the most important metrics in QoS routing, furthermore, many metrics (like delay , jitter) can be expressed as a function of the bandwidth [10].

2. Related Work

QoS routing has recently received a lot of attention and research has been published on many different aspects of the subject. General surveys can be found in [15, 4]. Although, localized QoS routing is a relatively new approach in the context of computer networks, the idea of routing using only local information has been used in many dynamic routing schemes proposed in the context of telephone networks, see [7] and references therein for more details. The most relevant work to ours is the proportional sticky routing (psr) algorithm proposed in [13] which will be used to compare the performance of our algorithm.

The basic idea of the psr scheme is to assume that the route-level statistics, such as the number of flows blocked, is the only available QoS state information at a source and based on these statistics, the algorithm attempts to proportionally distribute the load from a source to a destination among multiple paths according to their observed flow blocking probability. As mentioned previously, this algorithm requires every node maintain a predetermined set of candidate paths **R** to each possible destination. Since it has been shown that routing algorithms that prefer short paths usually outperform algorithms that do not take path length into consideration [18, 10]. The psr distinguishes between two types of paths, minhop paths R^{min} and alternative paths R^{alt}, where $R = R^{min} \bigcup R^{alt}$. This allows the algorithm to prefer minhop paths and limit the so-called "knock-on" effect which is the cascade effect that results from using alternative paths which in turn forces other sources that use these alternative paths as their minhop paths to use their alternative paths [13, 7]. The psr scheme can be viewed to operate in two stages: 1) proportional flow routing, and 2) computation of flow proportions. In the first stage, incoming flows are routed along paths selected from a set of eligible paths R^{elg}. A path r is selected from this set with a frequency determined by a prescribed proportion α_r. Initially, all the candidate paths are eligible and associated with a variable called the maximum permissible flow blocking parameter γ_r which determines the allowed number of blocked flows routed along this path before it becomes ineligible. For each minhop path, γ_r is set to $\hat{y}$ which is a configurable system parameter. For each alternative path, the value of γ_r is dynamically adjusted between 1 and $\hat{y}$. A cycle is completed when R^{elg} becomes empty and a new cycle is started with $R^{elg} = R$. An observation period consists of η cycles, at the end of each observation period; a new flow proportion α_r for each path $r \in R$ is computed based on its observed blocking probability b_r. Flow proportion for minhop paths are computed such that their flow blocking rates ($\alpha_r b_r$) are equal. The minimum blocking probability among the minhop paths b^* is used as the reference to control flow proportions for the alternative paths. That is, for each $r \in R^{alt}$, if $b_r < \psi\, b^*$, $\gamma_r = \min(\gamma_r + 1, \hat{y})$. 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 path to measure their quality.

Although, psr exhibits intelligence in routing decisions, the algorithm itself is based on a theoretical scheme that uses Erlang's loss formula to compute flow proportions which relies on the steady-state distribution of blocking probabilities. This introduces difficulties in calculating these flow proportions especially when dealing with Non-Poisson or bursty traffic. Furthermore, since the size of the candidate paths is shrinking during each cycle, the flow proportions computed at the beginning of the cycle need to be normalized every time a path becomes ineligible which means that the flow distribution may not reflect the predetermined proportions, since the last paths that become ineligible tend to receive more flows than others. Moreover, the method used by psr to limit the usage of non-minhop paths seems to be not very effective and simulations showed that psr algorithm tends to use non-minhop paths heavily, especially during low loads. While this may help to balance the load across the network, it is not a good idea to use long paths when it is possible to achieve the same performance with minhop paths.

3. The Proposed Algorithm (Credit Based Routing)

Unlike the psr scheme, which critically relies on the steady state of blocking probabilities to compute flow proportions and periodically adjusts them, Credit Based Routing (cbr) uses simple routing rules across the network, making constructive use of the inherent

```
Initialize
  set P.credits=MAX_CREDITS, $\forall$ P $\in$ R
CBR(MAX_CREDITS)
  1. if P.credits=0 $\forall$ P $\in$ R
       set P.credits=MAX_CREDITS, $\forall$ P $\in$ R
  2. $P^{min}$ = max{P.credits:P $\in$ $R^{min}$}.
  3. $P^{alt}$ = max{P.credits:P $\in$ $R^{alt}$}.
  4.1 if($P^{min}$.credits>=$\Phi \times P^{alt}$.credits)
       set P= $P^{min}$
  4.2 else
       set P= $P^{alt}$
  5. route flow along path P.
  6.1 if flow accepted
      UpdateBlockingProbability(P)
      amount=(1- P.BlockingProbability)
      P.credits=min{P.credits+amount,MAX_CREDITS}
  6.2 else
      UpdateBlockingProbability(P)
      amount=(P.BlockingProbability)
      P.credits=max{P.credits-amount,0}
```

Figuer 1. cbr pseudo code

randomness to search for good routing patterns. It monitors the flow blocking probabilities for each path and incorporates this information into a very simple crediting scheme that rewards a path upon flow acceptance and penalizes it upon flow rejection. The pseudo code for the proposed algorithm is shown in Figure 1.

Like psr, cbr requires every node to maintain a predetermined set of candidate paths **R** to each possible destination. For the same reasons mentioned when describing the psr scheme, cbr distinguishes between two types of paths, minhop paths set R^{min} and alternative paths set R^{alt}, where **R**= R^{min} U R^{alt}. Every path P is associated with a variable P.credits that stores the accumulated credits gained so far. Upon flow arrival, cbr selects two paths, P^{min} and P^{alt} which are the paths with maximum credits in R^{min} and R^{alt} respectively. The flow is routed along P^{min} if $P^{min}.\text{credits}>=\Phi \times P^{alt}.\text{credits}$, where $\Phi \leq 1$, otherwise, P^{alt} is chosen. Φ is a system parameter (analog to ψ in psr scheme) that controls the usage of alternative paths and limits the "knock-on" effect described earlier. We experimented with different values of Φ and found that setting Φ=1 gives good results, unless the average number of minhop paths $|R^{min}|$ is small compared to the average number of alternative paths $|R^{alt}|$, in this case we need to set Φ to a value between 0.85 and .95 to control the usage of the alternative paths. Note that, if there is more than one path with maximum credits, the first one is chosen. When the flow is accepted along the selected path, its blocking probability is updated accordingly and P.credits is incremented by an amount that corresponds to its success probability. On the other hand, if the flow is rejected, P.credits is decremented by an amount that corresponds to its blocking probability, as shown in the pseudo code.

It is possible to use a fixed crediting scheme whereby P.credits is incremented and decremented by a fixed amount. While this avoids the problem associated with computing blocking probabilities, simulations showed that incorporating blocking probabilities in the crediting scheme improves the performance, which is intuitively expected since paths with low blocking probabilities will receive more credits, hence, more flow will be routed along them and vice versa. Furthermore, fluctuations in P.credits will be reduced while flows are being accepted and rejected since credits are incremented or decremented gradually which enables P.credits to reflect the quality of the path more accurately. MAX_CREDITS is a system parameter which determines the maximum attainable credits for each path, i.e $0 \leq$ P.credits $\leq$ MAX_CREDITS . This has the following advantages. First, it stops a path with very low blocking probability from accumulating very large credits which could prevent the algorithm from selecting other good paths and causes undesirable behaviour. Second, if the quality of a path with very large credits degrades, it will take long time to bring its credits down to the correct level, hence, imposing this upper bound helps cbr to react and adapt in a timely manner. Third, it is more sensible to treat paths with credits beyond a certain level as if they have the same quality, even if some paths can attain more credits, this also ensures that the load will be balanced among these paths as their credits compete to reach this upper limit. Experimenting with different values of MAX_CREDITS, we found that values between 3 and 5 give good results and enables the algorithm to react in a timely manner.

Since cbr continuously monitors flow blocking probabilities, it records flow data (acceptance and rejection) for every path and uses a simple moving average with predetermined period to calculate its blocking probability. For a period of M, blocking probability of every path will be calculated using the most recent M flow data. For example , let s={$\underline{0}$,0,1,1,1} represent the data of the last M=5 flows where 0 indicate flow rejection and 1 indicates flow acceptance, then the blocking probability will be 2/5. Now, if another flow is accepted , then the oldest element will be deleted from s and replaced by the data from the last flow, i.e., s={1,0,1,1,1} and the blocking probability will be updated accordingly. In contrast to the psr scheme which needs relatively accurate estimations of blocking probabilities to compute flow proportions, cbr uses blocking probabilities merely to improve the crediting scheme hence it is not very sensitive to these blocking probabilities. Furthermore, we found by experiments that setting M to different values (10, 20, and 30) has insignificant impact on the performance of the cbr.

4. Performance Evaluation

In this section, we evaluate the performance of the cbr scheme and compare it with the psr scheme. As a reference, we also include in the comparison, the global QoS routing scheme widest shortest path (wsp) which searches for a feasible path with minimum hop count. If there are several such paths, the one with the maximum available bandwidth is chosen. We use the notation wsp(x) to refer to this algorithm with update interval of x time units. In the following, we first describe our simulation model and performance metrics and then compare the performance of the three schemes using these metrics.

4.1. Simulation Model

We developed a two-level simulator based on OMNeT++ [20], which is an event-driven simulator. At the flow-level, it selects routes based on a predetermined scheme (cbr, psr and wsp) and does admission control and resource reservation. The packet-level is developed to increase the level of realism by simulating connection setup and tear-down at the packet-level.

4.1.1 Network Topologies. A major challenge in studying routing algorithms in wide-area networks is how to represent the underlying topology given the dynamic nature of current networks. Poor understanding of the characteristics of these networks makes it very difficult to define a "typical" configuration [6]. Performance of routing algorithms may vary dramatically with the underlying network topology. Consequently, our simulation experiments consider a range of networks which represent different topologies with different characteristics. We study a familiar ISP topology (Figure 2) that has appeared in other routing studies [3, 16]. In addition, we investigate both random and regular topologies. The random topology was generated by Brite generator [11] using Waxman's model. Table 1 lists the most important characteristics of the topologies used in the experiments.

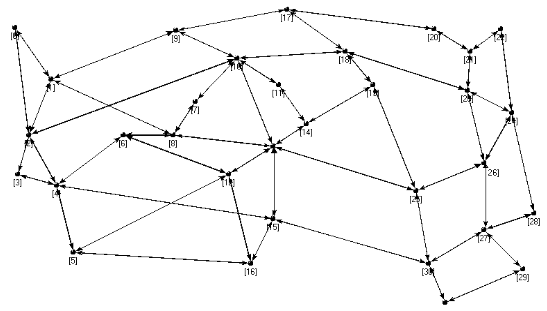


Figure 2. ISP topology

Table1. Topologies generated and their characteristics

Name	Nodes	Links	Avg. degree	Diam .	Avg. path length
RAND80	80	480	6	6	3.008
ISP	32	108	3.375	6	3.177
Torus	49	196	4	6	3.5

4.1.2 Simulation Setup. In all simulation experiments, links are assumed to be bidirectional and of the same capacity C in each direction (C=150Mbps). The topology remains fixed during each experiment, hence, link failures are not modeled. Flows arrive to each source node according to a Poisson process with rate λ and destination nodes are selected randomly. Each node is capable of being source and/or destination. Applying this uniform random selection of destinations, results in a uniform traffic in regular topologies, and non-uniform traffic in random and ISP topologies, this allows us to evaluate the performance under balanced and unbalanced loads.

Although, Poisson traffic is widely used to model network arrivals, recent studies [5] showed that the flow arrival process is bursty and that distribution with heavy tails such as Weibull, yield better models. Consequently, we also study the effect of bursty traffic where flows inter-arrival times follow Weibull distribution.

Flow duration is exponentially distributed with mean 1/μ, while flow bandwidths are uniformly distributed within two intervals, [0.1-2MB] and (2-4MB] for flow with small and large bandwidth requirements respectively. Following [18, 17], the offered network load is $\rho = \frac{\lambda N \overline{b} \overline{h}}{\mu L C}$, where N is the number of nodes, $\overline{b}$ is the average bandwidth required by a flow, $\overline{h}$ is the average path length (in number of hops) and L is the number of links in the network. Since the performance of routing algorithms may vary across different load conditions, our simulation experiments consider a wide rang of loads to assist in evaluating the algorithms under different load conditions. However, under low loads, flows are almost always accepted which results in very low blocking probabilities and all algorithms behave the same. Therefore, we try to chose the most relevant range of loads and omit the rest either because the relative performance of the three algorithms is reflected in the chosen range or the difference in the performance is insignificant.

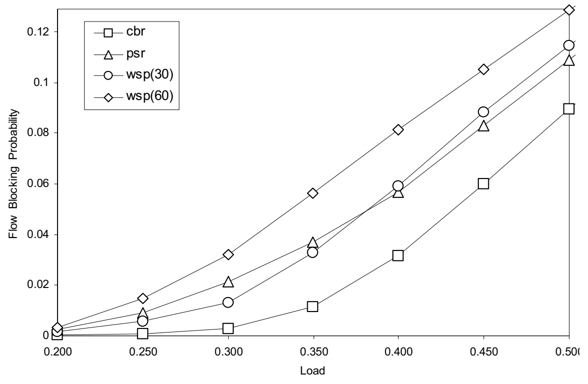
The parameters for the psr algorithm are η=3, ψ =0.8 and $\hat{\gamma}$=5 or $\hat{\gamma}$=10 (the one which gives the best results is chosen). For the cbr algorithm, MAX_CREDITS=5 and Φ=1 unless otherwise stated. Blocking probabilities are calculated based on the most recent 20 flows. Following [13], the set of candidate paths is chosen such that, for each source-destination pair, all the paths between them whose length is at most one hop more than the minimum number of hops are ncluded in the set. Each run simulates the arrival of 2,000,000 flows and the simulation results are collected after the first 500,000 flows. We use flow blocking probability as the main performance metric, which is defined as:

$$\text{Flow Blocking Probability}=\frac{|\boldsymbol{B}|}{|\boldsymbol{C}|}$$

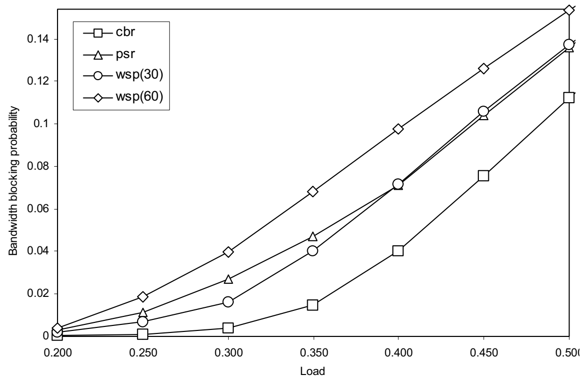
where **B** is the set of blocked flows and **C** is the set of the total flows. However, since large bandwidth requirements are hard to satisfy, flow blocking probability may not reflect the performance very accurately as it is possible to achieve lower flow blocking probability by discriminating against flows with large bandwidth requirements and preferring small bandwidth flows. Therefore, we use also the notion of bandwidth blocking probability which is defined as:

$$\text{Bandiwdth Blocking Probability}=\frac{\sum_{f\in \boldsymbol{B}} bandwidth(f)}{\sum_{f\in \boldsymbol{C}} bandwidth(f)}.$$

Obviously, if flows request the same amount of bandwidth then the two metrics will be the same.



(a) Flow blocking probability



(b) Bandwidth blocking probability

Figure 3. Blocking probabilities for RAND80.

4.2 Simulation Results

4.2.1 Flow blocking probabilities. Figure 3, compares the performance of the three algorithms in term of flow and bandwidth blocking probabilities under different load conditions. The overall flow and bandwidth blocking probabilities are plotted against the offered load using the random topology RAND80. We note the followings: First, under very low loads (load≤ 0.2), the difference in the performance (in terms of both metrics) of all the three algorithms is relatively small which is intuitively expected since the probability of finding sufficient available bandwidth in each link is high and flows are almost always accepted. However, performance varies significantly when the load increases, and the bandwidth blocking probability grows more rapidly than the flow blocking probability implying that flows with large bandwidth requirements are hard to route, as expected. Second, the relative performance of the three algorithms is the same for flow and bandwidth blocking probabilities suggesting that both metrics can be used to evaluate the performance of routing algorithms especially when the bandwidth requirements are small compared to links capacities. Third, the performance of the wsp scheme is significantly affected by the update period, for the wsp(60) we can see that it gives the worst performance and its blocking probability increases rapidly even under small and moderate loads. Forth, The psr performance is worse than the wsp(30) under low loads. However, as the load increases, the wsp(30) performance starts to degrade and the psr starts to outperform it, while the cbr scheme gives the best performance under all load conditions.

To see why this is, note that for the case of the wsp scheme, paths are selected based on a QoS global state which is updated periodically and when periodic updates do not respond quickly enough to variations in network resources, the performance will be affected significantly.

Another important factor is the effect of alternative routing (multi-paths routing). In one extreme, the wsp which always selects the best seemingly feasible path (based on its current global state) and sticks with it (even if it becomes not feasible afterwards) until the next new updates arrive to correct the situation. At the other extreme, the psr always tries to distribute the incoming flows across all paths in the candidate set according to their proportions and continues to do so until the end of the observation period where the proportion are adjusted again. cbr lies in the middle and selects the path with the maximum credits as long as it does not reject flows, however since credits of the selected path are updated after every flow routed along that path, any flow rejection will cause its credits to be

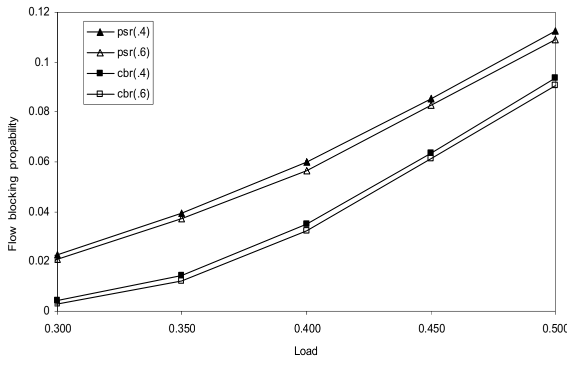


Fig.4 Impact of bursty traffic.

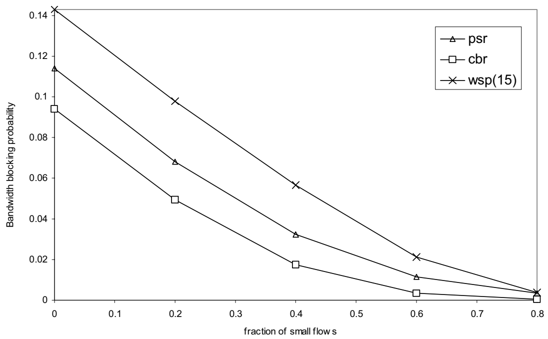


Fig. 5 Impact of large-bandwidth flows

decreased and an alternative path with more credits to be selected.

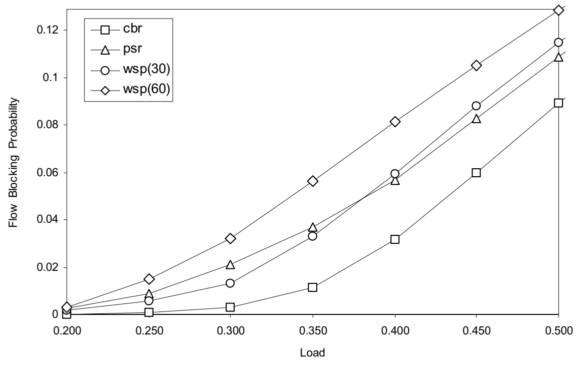
In the light of the above discussion, we can interpret the results in Fig.1 as follows: The wsp scheme with update interval 60 gives the worst performance due to its inability to react in a timely manner because of its long update period. However this algorithm is included here as a reference and more detailed analysis can be found in [18]. In the case of the psr algorithm, psr benefits from its multiple routing scheme and performs better than wsp(60), however, since the graph is very dense, more paths are available between each source/destination pair and paths in the candidate set are allowed to block more flows before the observation period ends and the proportions are adjusted. On the other hand, the cbr mechanism described above enables it to circumvent the problems associated with the other two schemes and performs well under varying load conditions.

4.2.2 Impact of bursty traffic. Following [18, 5], we model the burstiness of the traffic using the Weibull distribution with two different values of the shape parameter of the distribution (a), namely, a=0.4 and a=0.6 where burstiness is increased with a smaller shape value. Figure 4 shows the flow blocking probability plotted against the offered load with different shape values for the random topology RAND80. As we can see, increased burstiness in the arrival process results in increased blocking probability over the range of loads used. Although the impact of burstiness is not very significant in the RAND80 topology, other experiments results (not shown here) showed that burstiness has very significant effect on the performance of the two algorithms using different topologies.

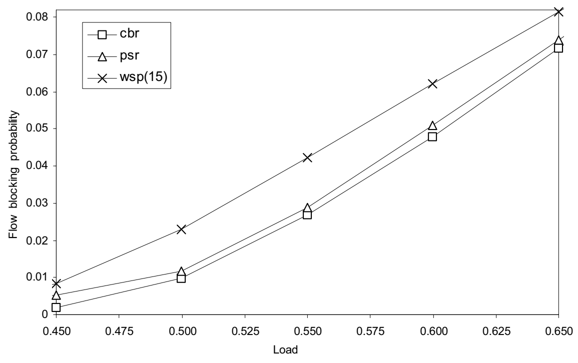
4.2.3 Impact of large-bandwidth flows. To study the effect of large bandwidth flows, we consider the case of a mixed traffic that contains two classes of flows, small-bandwidth flows and large-bandwidth flows. The amount of bandwidth requested by flows is chosen randomly from the range (0.1-2MB) for low-bandwidth flows and (2-4) for large-bandwidth flows. Both types have a flow duration which is exponentially distributed with mean equal 45 time units.

The performance is measured by varying the fraction of small-bandwidth flows. Figure 5 shows the flow blocking probability plotted against the fraction of small-bandwidth flows. As expected, as the fraction of small bandwidth increases, the blocking probability decreases and again, the cbr scheme gives the best performance. This shows that both cbr and psr perform well despite the fact that routing is independent of the amount of bandwidth requested. This remains true as long as the amount requested is small compared to link capacities. More details are given in [13].

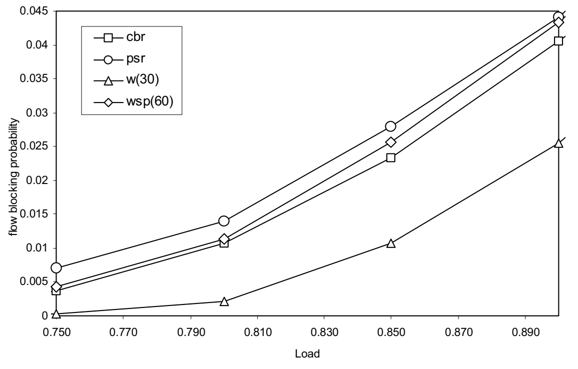
4.2.4 Impact of network topology. In view of the fact that the performance of any routing algorithm may vary dramatically with the underlying network topology, we evaluate the performance of our algorithm using different types of network topologies.



(a) Random topology



(b) ISP topology



(c) Torus topology

Figure 6. Impact of network topology.

However, our goal is not to provide a thorough evaluation of the impact of network topology on the performance of routing algorithms as this could be a future research direction, rather we aim to illustrate that our algorithm maintains its good performance across wide range of topologies. Figure 6 shows the flow blocking probability for the three schemes using the topologies described earlier in Table 1. From the figure we note the followings: The cbr scheme performs better than the psr scheme in all cases. In addition, we can see in case of the RAND80 and ISP topologies that our scheme gives superior performance and outperforms the wsp scheme even for relatively small update intervals (30 and 60). In fact, cbr performs better even for an update interval of 15 time unit (not shown in the graph).

In case of Torus topology, cbr still gives good results and its blocking probability is between that of wsp with update intervals of 60 and 30 which are still very small update intervals, since in practice, it is normal for the update interval to be as large as several minutes to reduce the overhead of distributing links states. However, in the Torus topology, both localized routing schemes are not able to provide comparable performance to the wsp scheme for small update intervals (≤ 30) which is most probably due to the fact that the traffic in this topology is uniform and the effect of route flapping exhibited by the wsp scheme is therefore reduced.

4.3 Complexity and overhead

Most global QoS routing algorithms use a variant of Dijkestra's or Belman-Ford's algorithms which take at least $O(N^2)$ time where N is the size of the network measured in number of nodes.

In contrast, cbr and psr schemes use simple method that involves the selection of the desired path among the set of the candidate paths R. In the psr scheme, the path is selected using use a weighted-round-robin-like path selector (wrrps) to pick a path from the current eligible path set and its worst case time complexity is $O(|R|)$. Similarly, cbr scheme chooses the path with the maximum credits among the set of the candidate paths which requires $O(|R|)$ operations in addition to updating blocking probabilities which takes constant time $O(1)$. In fact, cbr is simpler than the psr scheme, since no flow proportions calculation is involved. In contrast to the wsp scheme, both cbr and psr schemes share the advantage of requiring no global state to be maintained at each router. This is considered one of the main advantages of using localized routing schemes.

5. Conclusions

In this paper we presented a simple credits based localized algorithm for QoS routing that performs routing using only flow statistics collected locally. We compared its performance against the psr algorithm and we demonstrated through extensive simulations that our algorithm outperforms the psr algorithm. We also compared its performance against the wsp algorithm and showed that cbr gives a comparable performance with better time complexity and very low communication overhead which confirms the localized.

6. Acknowledgment

The authors would like to thank the two anonymous reviewers for their valuable comments and suggestion.

7. References

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

[2] S. Chen and K. Nahrstedt, *Distributed QoS Routing with Imprecise State Information*, *7th IEEE International Conference on Computer, Communications and Networks*, IEEE, Lafayette, LA, 1998, pp. 614-621.

[3] S. Chen and K. Nahrstedt, *Distributed Quality-of-Service routing in HighSpeed Networks Based on Selective Probing*, *23rd Annual Conference on Local Area Networks (LCN'98)*, IEEE, 1998, pp. 80--89.

[4] S. Chen and K. Nahrstedt, *An Overview of Quality-of-Service Routing for the Next Generation HighSpeed Networks: Problems and Solutions*, IEEE Network Magazine, 12 (1998), pp. 64 -79.

[5] A. Feldmann, *Characteristics of TCP connections arrivals*, in K. Park and W. Willinger, eds., *Self-similar Network Traffic and Performance Evaluation*, John Wiley and Sons, New York, 2000, pp. 367--399.

[6] S. Floyd and V. Paxson, *Difficulties in simulating the Internet*, IEEE/ACM Transations on Networking, 9 (2001), pp. 392-403.

[7] R. J. Gibbens, F. P. Kelly and P. B. Key, *Dynamic Alternative Routing*, in E. M. Steenstrup, ed., *Routing in Communications Networks*, Prentice Hall,, Englewood Cliffs, 1995, pp. 13-47.

[8] R. Guerin and A. Orda, *QoS-Based Routing in Networks with Inaccurate Information: Theory and Algorithms*, IEEE/ACM Transactions on Networking, 7 (1999), pp. 605-614.

[9] S. K. Kweon and K. G. Shin, *Distributed QoS routing using bounded flooding*, University of Michigan, 1999.

[10] Q. Ma and P. Steenkiste, *Quality-of-service routing for traffic with performance guarantees*, *5th Int. IFIP Workshop on QoS*, IEEE, New York, NY, 1997, pp. 115-126.

[11] A. Medina, A. Lakhina, I. Matta and J. Byers, *Brite: An approach to universal topology generation*, *Proceedings of the International Workshop on Modeling Analysis and Simulation of Computer and Telecommunications Systems*, Cincinnati, Ohio, 2001, pp. 346-356.

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

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

[14] H. D. Neve and P. V. Mieghem, *TAMCRA: a tunable accuracy multiple constraints routing algorithm*, Computer Communications Journal, 23 (2000), pp. 667—679.

[15] P. Paul and S. V. Raghavan, *Survey of qos routing*, *15th international conference on Computer communication*, Mumbai, India, 2002, pp. 50-75.

[16] H. Pung, J. Song and L. Jacob, *Fast and efficient flooding based QoS routing algorithm*, *IEEE ICCCN99*, 1999, pp. 298--303.

[17] A. Shaikh, J. Rexford and K. Shin, *Efficient Precomputation of Quality of Service Routes*, *International Workshop on Network and OS Support for Digital Audio and Video (NOSSDAV '98)*, IEEE, Cambridge, 1998, pp. 15-27.

[18] A. Shaikh, J. Rexford and K. Shin, *Evaluating the Impact of Stale Link State on Quality-of-Service Routing*, IEEE/ACM Transactions on Networking, 9 (2001), pp. 162--176.

[19] K. G. Shin, C.-C. Chou and S. K. Kweon, *Distributed Route Selection for Establishing Real-time Channels*, IEEE Transactions on Parallel and Distributed Systems, 11 (2000), pp. 318-- 335.

[20] A. Varga, *The OMNeT++ Discrete Event Simulation System*, *the European Simulation Multiconference*, Prague, Czech Republic, 2001.

[21] Z. Whang and J. Crowcroft, *Quality-of-Service routing for supporting multimedia applications*, IEEE J. Select. Areas Commun, 14 (1996), pp. 1228-1234.

[22] 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, pp. 102-107.