Adaptive Proportional Routing: A Localized QoS Routing Approach

Srihari Nelakuditi, Zhi-Li Zhang, and Rose P. Tsang

Abstract—**Most of the QoS routing schemes proposed so far require periodic exchange of QoS state information among routers, imposing both communication overhead on the network and processing overhead on core routers. Furthermore, stale QoS state information causes the performance of these QoS routing schemes to degrade drastically. In order to circumvent these problems, we focus on *localized* QoS routing schemes where the edge routers make routing decisions using only "local" information and thus reducing the overhead at core routers. We first describe *virtual capacity based routing* (vcr), a theoretical scheme based on the notion of *virtual capacity* of a route. We then propose *proportional sticky routing* (psr), an easily realizable approximation of *vcr* and analyze its performance. We demonstrate through extensive simulations that adaptive proportional routing is indeed a viable alternative to global QoS routing approach.**

I. Introduction

A. QoS Routing: Global vs. Localized Approaches

Quality-of-Service based routing, if done appropriately, can significantly improve network performance while providing support for QoS guarantees. The majority of QoS routing schemes [1], [3], [5], [10], [15], [17], [18] proposed so far require periodic exchange of *link QoS state* information among network routers to obtain a *global view of the network QoS state*. This approach to QoS routing is thus referred to as the *global* QoS routing approach. Because network resource availability changes with each flow arrival and departure, maintaining *accurate* network QoS state requires *frequent* information exchanges among the network nodes (routers). The prohibitive communication and processing overheads entailed by such frequent QoS state updates precludes the possibility of *always* providing each node with an *accurate* view of the current network QoS state. Consequently, *the network QoS state information acquired at a source node can quickly become out-of-date when the QoS state update interval is large relative to the flow dynamics*. Under these circumstances, exchanging QoS state information among network nodes is superfluous. Furthermore, path selection based on a *deterministic* algorithm such as Dijkstra's shortest path algorithm, where *stale QoS state information is treated as accurate*, does not seem to be judicious. In addition, the global view of the network QoS state may lead to the so-called *synchronization problem*: after one QoS state update, many source nodes choose paths with shared links because of their perceived available bandwidth, therefore causing over-utilization of these links. After the next QoS state update, the source nodes would avoid the paths with these shared links, resulting in their under-utilization. This oscillating behavior can have severe impact on the system performance, when the QoS state update interval is large. Due to these drawbacks, it has been shown that when the QoS update interval is large relative to the flow dynamics, the performance of global QoS routing schemes degrades drastically [1], [12], [15].

As a viable alternative to the global QoS routing schemes, in [12] we have proposed *a localized* approach to QoS routing. Under this approach, *no global QoS state information exchange among network nodes is needed.* Instead, source nodes 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. The proposed localized QoS routing approach has several advantages. First of all, without the need for global information exchange, the communication overhead involved is minimal. Second, core routers (i.e., non-source routers) do not need to keep and update any QoS state database necessary for global QoS routing, thereby reducing the processing and memory overhead at core routers. Last but not the least, the localized QoS routing approach does not require any modification or extension to existing routing protocols such as OSPF. Only source routers need to add a QoS routing enhancement to the existing routing module. This makes localized QoS routing schemes readily deployable with relatively low cost.

B. Adaptive Proportional Routing: A Localized Approach

The fundamental question in the design of localized QoS routing schemes is *how to perform path selection based solely on a local view of the network QoS state so as to minimize the chance of a flow being blocked as well as to maximize the overall system resource utilization.* The problem of path selection in localized QoS routing is complicated by many factors. For example, due to complex network topology, paths between many source-destinations pairs may have shared links whose capacity and load are unknown to the sources. Furthermore, the network load can fluctuate dynamically, which can make a previously unloaded link suddenly overloaded. In addition, path selection decision made by one source may affect the decision of another source.

To effectively address these difficulties, we study a novel *adaptive proportional routing* approach for designing localized QoS routing schemes. Here we assume that the *route-level* statistics, such as the number of flows blocked, is the only available QoS state information at a source. Based on these statistics, adaptive proportional routing attempts to proportionally distribute the load from a source to a destination among multiple paths according to their perceived quality (e.g., observed flow blocking probability). In other words, adaptive proportional routing exploits the inherent randomness in path selection by proportioning flows among multiple paths. This is fundamentally different from the conventional, *deterministic* path selection algorithms (e.g., Dijkstra shortest path algorithm) used in global routing schemes, which always choose the "best" feasible path to route a flow. As a result, adaptive proportional routing effectively avoids the synchronization problem associated with global QoS routing schemes.

There are three major objectives in our investigation of adaptive proportional routing: *adaptivity*, *stability* and *simplicity*. With only a localized view of the network QoS state, it is important to adjust flow proportions along various paths adaptively in response to the dynamically changing network load. Stability is essential to ensure efficient system resource utilization and thus

0-7803-5880-5/00/$10.00 (c) 2000 IEEE

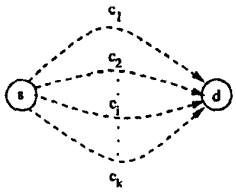


Fig. 1. A set of disjoint paths between a source and a destination

the overall flow throughput. Lastly, we are interested in employing *simple local rules and strategies at individual sources* to achieve adaptivity and ensure stability.

Towards these goals, we present a theoretical framework for studying adaptive proportional routing. Using Erlang's Loss Formula, we introduce the notion of *virtual capacity* which provides a mathematically sound means to model multiple paths between a source and a destination, as well as to compute flow proportions based on locally observed flow blocking probabilities. We also introduce a *self-refrained* alternative routing method to deal with the potential "knock-on" effect in QoS routing. By incorporating this *self-refrained* alternative routing method into the virtual capacity model, we design a theoretical adaptive proportional routing scheme which allows source nodes in a network to adaptively adjust their flow proportions based solely on locally observed flow blocking statistics. Through numerical examples we demonstrate the desired *self-adaptivity* of this theoretical adaptive proportional routing scheme in achieving an eventual equilibrium system state. As a simple and practical implementation of the theoretical scheme, we present a scheme, *proportional sticky routing* (psr), which preserves the self-adaptivity of the theoretical scheme while avoiding its computational overhead. Finally, comparison of the *psr* scheme with the well-studied global QoS routing scheme, the *widest shortest path* (wsp) scheme, is made using simulations. These simulation results demonstrate that with its low overhead and comparable performance, a simple and easy-to-implement localized QoS routing scheme such as *psr* provides a viable alternative to a global QoS routing scheme such as *wsp*.

The remainder of the paper is organized as follows. Section II presents a theoretical framework for studying adaptive proportional routing. Section III describes the *psr* scheme, and simulation results are shown in Section IV. Section V concludes the paper.

II. Adaptive Proportional Routing: A Theoretical Framework

A. An Idealized Proportional Routing Model

Consider a simple *fork* topology shown in Figure 1, where a source s and a destination d are connected by k *disjoint* paths $r_1, r_2, \ldots, r_k$. Each path r_i has a (bottleneck) capacity of c_i units of bandwidth, and is assumed to be known to the source s. Suppose flows arrive at the source s at an average rate λ, and the average flow holding time is μ. Throughout this section, we assume that flow arrivals are Poisson, and flow holding times are exponentially distributed. For simplicity, we also assume that each flow consumes 1 unit of bandwidth. In other words, path r_i can accommodate c_i flows at any time. *Without precise knowledge of the QoS state of a path (i.e., the available bandwidth of the path)*, a flow routed along the path has a certain probability of being blocked. Therefore, the question is how to route flows along these k paths so that the overall blocking probability is minimized. This problem can be formulated using the classic Erlang's Loss Formula as follows.

Suppose that, on the average, the proportion of flows routed along path r_i is α_i, where $i = 1, 2, \ldots, k$, and $\sum_{i=1}^{k} \alpha_i = 1$. Then the blocking probability b_i at path r_i is given by $b_i = E(\nu_i, c_i) = \frac{\frac{\nu_i^{c_i}}{c_i!}}{\sum_{n=0}^{c_i} \frac{\nu_i^n}{n!}}$, where $\nu_i = \alpha_i \frac{\lambda}{\mu}$ is referred to as the (average) load on path i. The total load on the system is denoted by $\nu = \sum_{i=1}^{k} \nu_i = \frac{\lambda}{\mu}$. To minimize the overall blocking probability, the *optimal* routing strategy (in the absence of precise knowledge of QoS state of each path) is therefore to route α_i^* proportion of flows along path r_i, $i = 1, 2, \ldots, k$, such that $\sum \alpha_i^* = 1$ and $\sum \nu \alpha_i^* b_i^*$ is minimized. This *optimal proportional routing* strategy can be implemented, for example, by routing flows to path r_i with probability α_i^*.

Given the total load ν and the path capacities c_i's, the optimal proportions α_i^*'s can be computed using an iterative search technique (e.g., hill-climbing) starting with a set of arbitrary proportions. For $k > 2$, the procedure of computing the optimal proportions is generally quite complex to implement in practice. To circumvent this problem, we consider two alternative strategies for flow proportioning: *equalizing blocking probability* (ebp) and *equalizing blocking rate* (ebr). The objective of the *ebp* strategy is to find a set of proportions $\{\tilde{\alpha}_1, \tilde{\alpha}_2, \ldots, \tilde{\alpha}_k\}$ such that flow blocking probabilities of all the paths are equalized, i.e., $\tilde{b}_1 = \tilde{b}_2 = \cdots = \tilde{b}_k$, where $\tilde{b}_i$ is the flow blocking probability of path r_i, and is given by $E(\tilde{\alpha}_i \nu, c_i)$. On the other hand, the objective of the *ebr* strategy is to find a set of proportions $\{\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_k\}$ such that *flow blocking rates* of all the paths are equalized, i.e., $\hat{\alpha}_1 \hat{b}_1 = \hat{\alpha}_2 \hat{b}_2 = \cdots = \hat{\alpha}_k \hat{b}_k$, where $\hat{b}_i$ is the flow blocking probability of path r_i, and is given by $E(\hat{\alpha}_i \nu, c_i)$.

Unlike the optimal proportions, α_i^*'s, the proportions of *ebp*, $\tilde{\alpha}_i$'s, and those of *ebr*, $\hat{\alpha}_i$'s, can be computed using a simple iterative procedure starting with any arbitrary proportions. For example, consider the *ebp* strategy. Suppose we start with an initial set of proportions $\alpha_1^{(0)}, \alpha_2^{(0)}, \cdots, \alpha_k^{(0)}$. Let the corresponding blocking probabilities be $b_1^{(0)}, b_2^{(0)}, \cdots, b_k^{(0)}$, where $b_i^{(0)} = E(\alpha_i^{(0)} \nu, c_i)$. If $b_i^{(0)}$'s are all equal, then $\alpha_i^{(0)}$'s are the desired proportions. Otherwise, we use the mean blocking probability over all the paths, $\bar{b}^{(0)} = \sum b_i^{(0)} / k$, as the target blocking probability for each path, and obtain a new set of proportions, $\alpha_i^{(1)}$'s. The new proportions $\alpha_i^{(1)}$'s are computed from the Erlang's Loss Formula as follows: for $i = 1, 2, \ldots, k$, find the new load on path r_i, $\nu_i^{(1)}$, such that $\bar{b}^{(0)} = E(\nu^{(1)_i}, c_i)$. Then $\alpha_i^{(1)} = \frac{\nu_i^{(1)}}{\sum_{j=1}^{k} \nu_j^{(1)}}$. This procedure is repeated iteratively until we obtain a set of proportions such that the corresponding blocking probabilities are equal. Since for a fixed c_i the blocking probability b_i is an increasing function of its load $\alpha_i \nu$, it can be shown that the above iterative procedure will always converge. In the case of the *ebr* strategy, a similar iterative procedure can

TABLE I

COMPARISON OF *ebp*, *ebr* AND *opr*

Scenario		*ebp*	*ebr*	*opr*
$c_1 = 20, c_2 = 20, \nu = 27$	*obp*	2.36	2.36	2.36
$c_1 = 10, c_2 = 20, \nu = 22$	α_1	0.292	0.327	0.307
	α_2	0.708	0.673	0.693
	b_1	5.68	8.68	6.87
	b_2	5.62	4.22	5.01
	obp	5.64	5.68	5.58
$c_1 = 10, c_2 = 20, \nu = 18$	α_1	0.277	0.307	0.290
	α_2	0.723	0.693	0.710
	b_1	1.82	3.01	2.29
	b_2	1.82	1.33	1.59
	obp	1.82	1.84	1.80

be used to obtain a set of proportions which equalize the blocking rates of all the paths.

Table I shows the convergence points of the *ebp*, *ebr*, and *opr* strategies for a source and destination pair with two disjoint paths under different scenarios. As expected, when the capacities are equal all three strategies assign equal proportions for the two paths and yield same overall blocking probability. However, when the capacities are not equal, the equilibrium proportions for the two paths under the three strategies are different. It can be observed, however, the overall blocking probabilities under the *ebp* and *ebr* strategies are both quite close to that of the optimal strategy. Since it is generally computationally cumbersome to find the optimal equilibrium proportions, in this paper we explore the two simple strategies, *ebp* and *ebr*, for adaptive proportional routing.

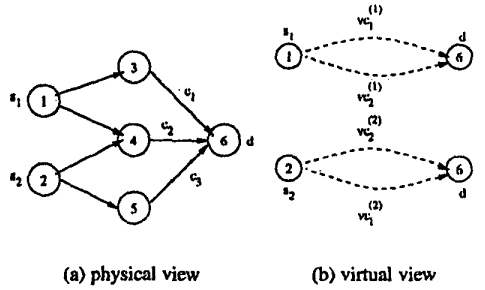
B. Virtual Capacity Model

In the idealized proportional routing model described above, we have assumed that all paths between a source and a destination are disjoint and their bottleneck link capacities are known. In practice, however, paths between a source and a destination have shared links. These paths may also share links with paths between other source-destination pairs. Furthermore, as traffic patterns across a network change, the bottleneck link of a path and its (perceived) capacity may also change. In order to address these issues, we introduce the notion of *virtual capacity* (vc) of a path.

Consider a source-destination pair. We model each path between them as one direct *virtual link* with a certain amount of capacity, referred to as the *virtual capacity* of the path. This virtual capacity is a function of *the load offered by the source along the path and the corresponding blocking probability observed by the source.* Formally, consider a path r between a source and a destination. Suppose a load of ν_r is offered by the source along the path, and the corresponding blocking probability observed by the source is b_r. Then the virtual capacity of the path, denoted by vc_r, is given by $vc_r = E_{vc}^{-1}(\nu_r, b_r)$, where $E_{vc}^{-1}(\nu_r, b_r)$ denotes the inverse function of the Erlang's Loss Formula with respect to the capacity, and is given by

$$vc_r = E_{vc}^{-1}(\nu_r, b_r) := \min\{c \geq 0 : E(\nu_r, c) \leq b_r\}$$

The notion of virtual capacity provides a mathematically sound way to deal with shared links among multiple paths. For example, suppose m paths, $r_1, r_2, \ldots, r_m$,share a *bottleneck* link with capacity c. Then the virtual capacity vc_i of path r_i represents its "capacity share" of the bottleneck link. Let ν_i denote the offered load on the bottleneck link from path r_i. Then the blocking probability on the bottleneck link is given by $b = E(\sum_{i=1}^{m} \nu_i, c)$. Since flows routed along any of these m paths have the same probability to be blocked at the bottleneck link, the virtual capacity of path r_i is given by $vc_i = E_{vc}^{-1}(\nu_i, b_i) = E_{vc}^{-1}(\nu_i, b)$, where b_i denotes the observed blocking probability of path r_i. In particular, for path r_i, the larger the offered load ν_i is, the larger is its virtual capacity vc_i. This reflects the larger "capacity share" of the bottleneck obtained by path r_i because of its higher offered load [1] .



(a) physical view (b) virtual view

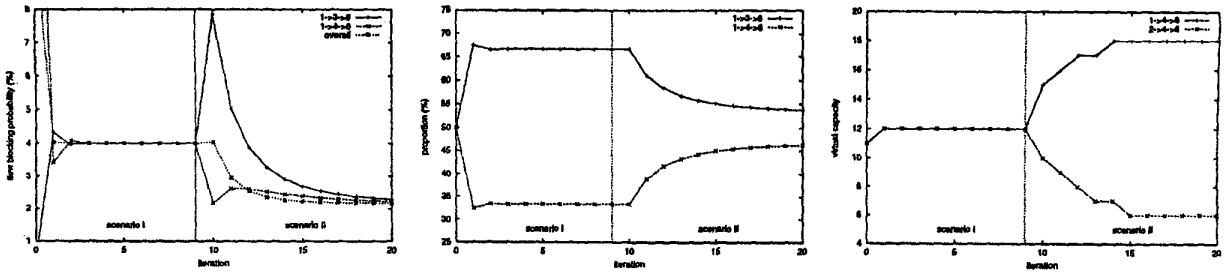
Fig. 2. Illustration of virtual capacity model using *kite* topology

Based on this notion of virtual capacity, we can model paths between a source and a destination as if they were all disjoint and had bottleneck capacities equal to their virtual capacities, as in the idealized proportional routing model (Figure 1). Unlike the idealized proportional routing model, however, the virtual capacity of a path is not fixed, but is a function of its offered load and the corresponding blocking probability. Since the virtual capacity of a path depends only on local statistics at a source (i.e., the offered load by a source and the corresponding blocking probability observed by the source), *flow proportioning based on virtual capacities of paths does not require any global QoS state information exchange.*

A key feature of our virtual capacity model is its *self-adaptivity*: proportions of flows (and therefore offered loads) along different paths between a source and a destination will be adjusted based on the observed blocking probability of those paths, an important measure of the "quality" of a path. From the definition of virtual capacity, we observe that for two paths with the same offered load, the path with higher observed blocking probability has lower virtual capacity. Therefore, if we are to equalize the observed blocking probabilities or blocking rates along these two paths, more flows should be routed to the path with lower observed blocking probability (and higher virtual capacity). The new proportions for these two paths can be computed based on their virtual capacities, as in the idealized proportional routing model.

We illustrate the self-adaptivity of the virtual capacity model through an example. Consider the *kite* topology shown in Figure 2(a), where two sources, s_1 and s_2, have two paths each to

[1] It is also worth noting that $\sum_{i=1}^{m} vc_i \geq c$. This is due to "loss in multiplexing gain" when a shared channel is divided into multiple "dedicated" channels. To ensure the same blocking probability, the total capacity of the dedicated channels has to be larger than the capacity of the shared channel



(a) *changes in blocking probabilities* (b) *changes in proportions* (c) *changes in vc's*

Fig. 3. Adaptation process of *ebp*

destination d, and two of the paths share a common link ($4 \rightarrow 6$). The links with labels are the bottleneck links of the network, where $c_1 = c_2 = c_3 = 20$, and all the other links can be viewed to have infinite capacities (i.e., flows are never blocked on these links). Let r_1^1, r_2^1 denote the paths $1 \rightarrow 3 \rightarrow 6$ and $1 \rightarrow 4 \rightarrow 6$ respectively, and r_1^2, r_2^2 denote the paths $2 \rightarrow 5 \rightarrow 6$ and $2 \rightarrow 4 \rightarrow 6$ respectively. The *virtual capacity view* of the two source-destination pairs are shown in Figure 2(b), where the paths r_2^1 and r_2^2 appear to each source as if they were disjoint with capacities vc_2^1 and vc_2^2 respectively. Note that if a path doesn't share links with any other path, its virtual capacity is the same as its actual bottleneck link capacity.

First consider the scenario where both sources have an offered load of 22. Suppose initially each source proportions flows *equally* between its two paths, i.e., $\nu_j^i = 11$, $i, j = 1, 2$. The blocking probabilities observed on paths r_1^1, r_2^1, r_2^2 and r_1^2 are $b_1^1 = 0.0046$, $b_2^1 = 0.2090$, $b_2^2 = 0.2090$, and $b_1^2 = 0.0046$ respectively, resulting in an overall blocking probability of 0.1068. The corresponding virtual capacities are $vc_1^1 = 20$, $vc_2^1 = 12$, $vc_2^2 = 12$, and $vc_1^2 = 20$. In particular, we see that the shared link of paths r_2^1 and r_2^2 is treated by each source as an exclusive link with capacity 12. For both sources, since the blocking probability of path r_2^i is much higher than path r_1^i, more flows will be proportioned to path r_1^i, as it has a larger virtual capacity vc_1^i. The new proportions can be computed based on the virtual capacities of the paths, using either the *ebp* strategy or the *ebr* strategy. For example, using the *ebp* strategy, the adaptation process for source s_1 is shown on the *left* side (scenario I) of Figure 3(a), where we see that after a few iterations the flow blocking probabilities of both paths r_1^1 ($1 \rightarrow 3 \rightarrow 6$) and r_2^1 ($1 \rightarrow 4 \rightarrow 6$) are equalized at around 0.04. Figure 3(b) shows the corresponding proportions of flows routed along these two paths during this adaptation process, where we see that source s_1 backs off from the path (r_2^1) with the shared bottleneck link $4 \rightarrow 6$, and directs more flows to the other path (r_1^1). The resulting flow proportions for path r_1^1 and r_2^1 at the *equilibrium* state are respectively 0.667 and 0.333.

Now consider the scenario where after the above equilibrium state is achieved, the offered load at s_1 increases from 22 to 25 whereas the offered load at s_2 decreases from 22 to 15. Given the new load at both sources, routing flows along the paths using the old equilibrium proportions no longer results in an equilibrium state. In particular, source s_1 sees a blocking probability of $b_1^1 = 0.0784$ on path r_1^1 and a blocking probability of $b_2^1 = 0.0216$ on path r_2^1. On the other hand, source s_2 sees a blocking probability of $b_1^2 = 0.0018$ on path r_1^2 and a blocking probability of $b_2^2 = 0.0216$ on path r_2^2. Hence, in an effort to equalize the blocking probabilities on both paths, s_1 will direct more flows to path r_2^1 and s_2 will direct more flows to path r_1^2. The new adaptation process is shown on the right side (scenario II, starting with iteration 10) of Figure 3(a). From the figure we see that as source s_1 directs more flows to path r_2^1, the observed blocking probability on path r_2^1 gradually increases while the observed blocking probability on path r_1^1 gradually decreases. These two blocking probabilities are eventually equalized at around 0.022. The proportions of flows routed along the two paths by source s_1 during this adaptation process are shown in Figure 3(b), where the equilibrium flow proportions for paths r_1^1 and r_2^1 are around 0.537 and 0.463, respectively.

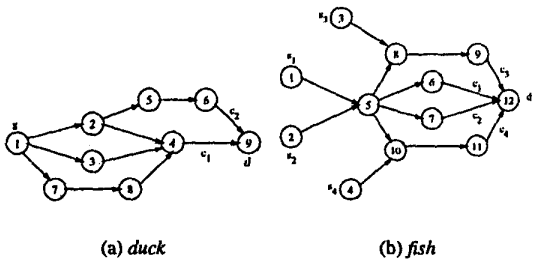
It is interesting to note that each source adapts to the load changes *not* with any *global* objective *but* with a *local* objective of equalizing blocking probabilities or rates among all paths to a given destination. This in turn results in an overall near-optimal stable system performance. For example, in scenario I, both source s_1 and source s_2 have an equal capacity share on the bottleneck link $4 \rightarrow 6$, each with a virtual capacity of 12. But as the load changes at each source, source s_1 starts routing more flows to path r_2^1, whereas source s_2 starts backing off from the path r_2^2, thereby allowing s_1 to grab more capacity share on the bottleneck link. The changes in the virtual capacity of the shared link seen by each source are shown in Figure 3(c). At the end, source s_1 has a virtual capacity of 18 from the shared bottleneck link, while source s_3 has a virtual capacity of 6. Due to this change in capacity shares, the blocking probability observed by source s_1 is reduced from 0.0595 at the onset of load change to 0.0225 in the end while that of s_2 goes up from 0.0084 to 0.0202. However, as a consequence of these self-adaptations at the two sources, the overall *system* blocking probability is reduced from 0.0404 to 0.022 (Figure 3(a)).

C. Self-Refrained Alternative Routing

In the virtual capacity model all paths between a source and a destination are treated equally. Since an admitted flow consumes bandwidth and buffer resources at all the links along a path,

1. PROCEDURE VCR()
2. Set $\bar{\beta}^{(n)} = \frac{\sum_{r \in R^{min}} \alpha_r^{(n)} b_r^{(n)}}{k^{min}}$
3. Set $b^* = \min_{r \in R^{min}} b_r^{(n)}$
4. For each path $r \in R$
5. Compute virtual capacity $vc_r^{(n)} = E_{vc}^{-1}(\nu_r^{(n)}, b_r^{(n)})$
6. For each minhop path $r \in R^{min}$
7. Compute target load ν_r^* such that $\bar{\beta}^{(n)} = \nu_r^* E(\nu_r^*, vc_r^{(n)})$
8. For each alternative path $r \in R^{alt}$
9. Compute target load ν_r^* such that $\psi b^* = E(\nu_r^*, vc_r^{(n)})$
10. For each path $r \in R$
11. Compute new proportion $\alpha_r^{(n+1)} = \frac{\nu_r^*}{\sum_{r \in R} \nu_r^*}$
12. END PROCEDURE

Fig. 4. The *vcr* procedure



(a) *duck* (b) *fish*

Fig. 5. Topologies used for illustration

clearly path length is also an important factor that we must take into consideration. As a general principle, it is always preferable to route a flow along *minhop* (i.e. shortest) paths than paths of longer length (also referred to as *alternative* paths). By preferring minhop paths and discriminating against alternative paths, we not only reduce the overall resource usage but also limit the so-called "knock-on" effect [7], [8], thereby ensuring the stability of the whole system. The "knock-on" effect refers to the phenomenon where using alternative paths by some sources force other sources whose minhop paths share links with these alternative paths to also use alternative paths. This cascading effect can result in a drastic reduction of the overall resource utilization of the network. In order to deal with the "knock-on" effect, we introduce a *self-refrained* alternative routing method, which provides an adaptive way to discriminate against "bad" alternative paths *without explicit trunk reservation* [8].

Consider a source-destination pair. Suppose there are k^{min} number of minhop paths between this source-destination pair, and let R^{min} denote the set of these minhop paths. The set of alternative paths is denoted by R^{alt}. Thus the set of all feasible paths $R = R^{min} \cup R^{alt}$. The basic idea behind the *self-refrained* alternative routing method is to ensure that *an alternative path is used to route flows between the source-destination pair only if it has a "better quality" (measured in flow blocking probability) than any of the minhop paths.* Formally, for a path $r \in R^{min}$, let b_r denote the observed flow blocking probability on path r. The minimum flow blocking probability of all the minhop paths, $b^* = \min_{r \in R^{min}} b_r$, is used as the reference in deciding a target flow blocking probability for alternative paths. The target flow blocking for alternative paths is set to ψb^*, where ψ is a configurable parameter to limit the "knock-on" effect under system overloads. An alternative path $r' \in R^{alt}$ is selected to route flows only if it can *attain* the target flow blocking probability. In other words, its observed flow blocking probability $b_{r'}$ is less than or equal to ψb^*.

This *self-refrained* alternative routing method has several attractive features. By using b^* as the reference in determining a target flow blocking probability for alternative paths, it dynamically controls the extent of alternative routing according to both the load at the source and the overall system load. For example, if both the load at the source and the overall system load is light, use of alternative paths will be kept at a minimum. However, if the load at the source is heavy but the overall system load is light, more alternative routes will be used by the source. Furthermore, by using only those alternative paths whose observed blocking probabilities are at most as high as the minimum of those of the minhop paths, we guarantee that the minhop paths are preferred to alternative paths. In particular, if an alternative path of a source-destination shares a bottleneck with one of its minhop paths, this alternative path is automatically "pruned". In addition, a source would gradually back off from an alternative path once its observed flow blocking probability starts increasing, thereby adapting gracefully to the change in the network load.

D. Virtual Capacity based Routing

By incorporating this *self-refrained* alternative routing method into the virtual capacity model, we devise a theoretical adaptive proportional routing scheme, which is referred to as the *Virtual Capacity based Routing* (vcr) scheme. In this *vcr* scheme, we use the *ebr*[2] strategy to proportion flows along the minhop paths, whereas proportions of flows along the alternative paths are computed using the target flow blocking probability ψb^*, as in the *self-refrained* alternative routing method. The scheme is shown in Figure 4. Suppose the total load for a source-destination pair is ν. At a given step $n \geq 0$, let $\nu_r^{(n)} = \alpha_r^{(n)} \nu$ be the amount of the load currently routed along a path $r \in R$, and let $b_r^{(n)}$ be its observed blocking probability on the path. Then the virtual capacity of path r is given by $vc_r = E_{vc}^{-1}(\nu_r^{(n)}, b_r^{(n)})$ (line 5). For each minhop path, the mean blocking rate of all the minhop paths, $\bar{\beta}^{(n)}$, is used to compute a new target load (lines 6-7). Similarly, for each alternative path, a new target load is determined using the target blocking probability ψb^* (lines 8-9). Given these new target loads for all the paths, the new proportion of flows, $\alpha_r^{(n+1)}$, for each path r is obtained in lines 10-11, resulting in a new load $\nu_r^{(n+1)} = \alpha_r^{(n+1)} \nu$ on path r.

In the following we illustrate through numerical examples how the vcr scheme uses alternative paths in a judicious and self-adaptive manner. Consider the *duck* topology shown in Figure 5(a). Let r_1^{min} and r_2^{min} denote, respectively, the two minhop paths $1 \to 2 \to 4 \to 9$ and $1 \to 3 \to 4 \to 9$. Similarly let r_3^{alt}, and r_4^{alt} denote, respectively, the two alternative paths $1 \to 2 \to 5 \to 6 \to 9$ and $1 \to 7 \to 8 \to 4 \to 9$. The two minhops r_1^{min} and r_2^{min} share the bottleneck link $4 \to 9$ with the alternative path r_4^{alt}. On the other hand, the minhop path r_1^{min} and the alternative path r_3^{alt} share the link $1 \to 2$, which

[2]We found *ebr* strategy to be more amenable than *ebp* strategy for implementation.

```
1.  PROCEDURE PSR-ROUTE()
2.    Select an eligible path r = wrrps(R_E)
3.    Increment flow counter, n_r = n_r + 1
4.    If failed to setup connection along r
5.      Decrement failure counter, f_r = f_r - 1
6.      If failures reached limit, f_r == 0
7.        Remove r from eligible set, R_E = R_E - r
8.        If eligible set is empty, R_E == ∅
9.          Reset eligible set, R_E = R
10.         For each path r ∈ R
11.           Reset failure counter, f_r = γ_r
12. END PROCEDURE
```

```
1.  PROCEDURE PSR-PROPO-COMPU()
2.    For each path r ∈ R
3.      Compute blocking probability, b_r = nγ_r / n_r
4.      Assign a proportion, α_r = n_r / Σ_{r*∈R} n_{r*}
5.    Set target blocking probability, b* = min_{r∈R^min} b_r
6.    For each alternative path r' ∈ R^alt
7.      If blocking probability high, b_r' >= b*
8.        Decrement failure limit, γ_r' = γ_r' - 1
9.      If blocking probability low, b_r' < ψb*
10.       Increment failure limit, γ_r' = γ_r' + 1
11. END PROCEDURE
```

Fig. 6. The *psr* procedure: (a) proportional routing and (b) proportion computation

is *not* a bottleneck link. The capacities c_1 and c_2 of bottleneck links are set to 20. Assume that a load of 30 is offered at source s. With the ψ parameter set to 0.8 and starting with a set of arbitrary proportions for the four paths, the *vcr* scheme would eventually reach a set of equilibrium proportions, which are $\alpha_1^{min} = 0.255$, $\alpha_2^{min} = 0.255$, $\alpha_3^{alt} = 0.490$ and $\alpha_4^{alt} = 0.000$, respectively. We see that a total 51% of the flows are routed through the bottleneck link $4 \rightarrow 9$. This link is shared equally by the two minhop paths, r_1^{min} and r_2^{min}, each with a blocking probability of 0.0508. The alternative path, r_3^{alt}, which also share the bottleneck link with the two minhop paths, is effectively cut off from the link and not used at all. This is because routing any flows through r_3^{alt} would only increase the resource usage without resulting in any decrease in the overall blocking probability. In contrast, the alternative path, r_3^1, is used to route 47% of the flows, with a blocking probability of 0.0406, which matches the target blocking probability for the alternative paths, $\psi b^* = 0.0406$. Since r_3^{alt} shares a *non-bottleneck* link with r_1^{min}, routing flows through r_3^{alt} helps reduce the overall blocking probability. Simulations using the *fish* topology (shown in Figure 5(b)) also demonstrate that the *vcr* scheme can adaptively respond to the traffic load changes along the alternative paths by adjusting the proportion of flows routed along these paths. These results are not included here due to space limitations, but can be found in [13]. These results validate that, with the *self-refrained* alternative routing method, a source judiciously chooses alternative paths to route flows *without actually being aware of where the bottleneck links are.*

III. Proportional Sticky Routing: An Implementation of VCR

In the previous section we presented an analytical framework for studying adaptive proportional routing. In particular, based on this framework we described a theoretical adaptive routing scheme — the *vcr* scheme, and demonstrated its self-adaptivity through several numerical examples. There are two difficulties involved in implementing the virtual capacity model. First, computation of virtual capacity and target load using Erlang's Loss Formula can be quite cumbersome. Second, and perhaps more importantly, the accuracy in using Erlang's Loss Formula to compute virtual capacity and new load relies critically on steady-state observation of flow blocking probability. Hence small statistic variations may lead to erroneous flow proportioning, causing undesirable load fluctuations. In order to circumvent these difficulties, we are interested in a simple yet robust

$$r_1\ r_2\ r_1\ r_3\ r_1\ r_2\ r_1\ r_4\ r_1\ r_2\ r_1\ r_3$$

Fig. 7. A sample *wrr* sequence for paths with weights $\alpha_{r_1} = 1/2$, $\alpha_{r_2} = 1/4$, $\alpha_{r_3} = 1/8$, $\alpha_{r_4} = 1/8$. This sequence has the property that in every window of size 2 there is an r_1 and an r_2 in every window of size 4. Similarly one r_3 and one r_4 in all windows of size 8. Assuming that up to the last r_1 are the paths chosen so far, the next path selected on the fly by the *wrr* path selector would be r_3.

implementation of the *vcr* scheme. In this section we present such an implementation which we refer to as the *Proportional Sticky Routing* (psr) scheme.

The *psr* scheme can be viewed to operate in two stages: 1) proportional flow routing, and 2) computation of flow proportions. The proportional flow routing stage proceeds in *cycles* of variable length. During each cycle incoming flows are routed along paths selected from a set of eligible paths. A path is selected with a frequency determined by a prescribed proportion. A number of cycles form an *observation period*, at the end of which a new flow proportion for each path is computed based on its observed blocking probability. This is the computation of flow proportion stage. As in the *vcr* scheme, flow proportions for minhop paths of a source-destination pair are determined using the *ebr* strategy, whereas flow proportions for alternative paths are determined using a target blocking probability. In the following we will describe these two stages in more detail.

Given an arbitrary source-destination pair, let R be the set of feasible paths between the source-destination pair, where $R = R^{min} \cup R^{alt}$. We associate with each path $r \in R$, a *maximum permissible flow blocking* parameter γ_r and a corresponding *flow blocking counter* f_r. For each minhop path $r \in R^{min}$, $\gamma_r = \hat{\gamma}$, where $\hat{\gamma}$ is a configurable system parameter. For each alternative path $r' \in R^{alt}$, the value of $\gamma_{r'}$ is dynamically adjusted between 1 and $\hat{\gamma}$, as will be explained later. As shown in Figure 6(a), at the beginning of each cycle, f_r is set to γ_r. Every time a flow routed along path r is blocked, f_r is decremented. When f_r reaches zero, path r is considered *ineligible*. At any time only the set of *eligible* paths, denoted by R_E, is used to route flows. Once R_E becomes empty, the current cycle is ended and a new cycle is started with $R_E = R$ and $f_r = \gamma_r$.

For an incoming flow, we use a weighted-round-robin-like path selector (*wrrps*) to pick a path from the current eligible path set R_E to route the flow. For a given set R_E of eligible paths and their associated proportions $\{\alpha_r, r \in R_E\}$, *wrrps* picks a path $r \in R_E$ based on its weight, $w_r = \frac{\alpha_r}{\sum_{s \in R_E} \alpha_s}$. This is implemented by generating a sequence of paths that preserves

TABLE II

COMPARISON OF PROPORTIONING IN *vcr* AND *psr*

Topo	Scenario		*vcr*	*psr*
kite	$\nu_{s_1} = 22, \nu_{s_2} = 22$	$\alpha_{1\to4\to6}$	0.356	0.351
		$\alpha_{2\to4\to6}$	0.356	0.357
		overall blocking	4.15%	4.24%
kite	$\nu_{s_1} = 25, \nu_{s_2} = 15$	$\alpha_{1\to4\to6}$	0.447	0.455
		$\alpha_{2\to4\to6}$	0.208	0.193
		overall blocking	2.45%	2.12%
duck	$\nu_s = 30$	$\alpha_{1\to2\to4\to9}$	0.255	0.269
		$\alpha_{1\to3\to4\to9}$	0.255	0.236
		$\alpha_{1\to2\to5\to6\to9}$	0.490	0.474
		$\alpha_{1\to7\to8\to4\to9}$	0.000	0.021
		overall blocking	4.58%	4.57%

TABLE III

COMPARISON OF BLOCKING (%) IN *psr* AND *srr*

Topo	Scenario	*vcr*	*psr*	*srr*
fork (3)	$\nu_s = 45$	4.56	5.12	7.67
fork (3)	$c_2 = 10, c_3 = 5, \nu_s = 25$	6.53	6.38	9.72
kite	$\nu_{s_1} = 22, \nu_{s_2} = 22$	4.15	4.24	5.59
kite	$\nu_{s_1} = 25, \nu_{s_2} = 15$	2.45	2.12	3.58
duck	$\nu_s = 30$	4.58	4.57	9.13
fish	$\nu_{s_1} = 20, \nu_{s_2} = 15$ $\nu_{s_3} = 5, \nu_{s_4} = 10$	1.36	1.02	4.40

flow proportions within as small a window as possible. This sequence is generated by *wrrps* on the fly: for an incoming flow, *wrrps* generates the next path in the sequence and routes the flow along the path. A sample *wrrps* sequence is shown in Figure 7, where the current eligible set R_E has four paths r_1, r_2, r_3 and r_4. Due to the space limitations, we omit the details regarding how *wrrps* generates such a sequence for a given set of eligible paths.

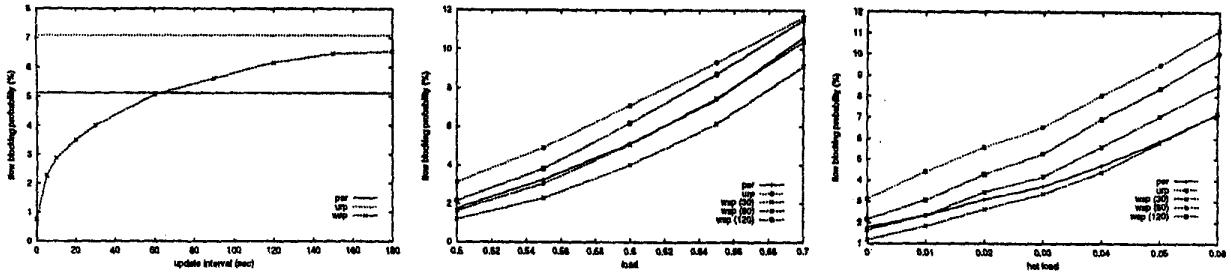
Flow proportions $\{\alpha_r, r \in R\}$ are recomputed at the end of each observation period (see Figure 6(b)). An observation period consists of η cycles, where η is a configurable system parameter used to control the robustness and stability of flow statistics measurement. During each observation period, we keep track of the number of flows routed along each path $r \in R$ using a counter n_r. At the beginning of an observation period, n_r is set to 0. Every time path r is used to route a flow, n_r is incremented. Since an observation period consists of η cycles, and in every cycle, each path r has exactly γ_r flows blocked, the observed flow blocking probability on path r is $b_r = \frac{\eta\gamma_r}{n_r}$. For each minhop path $r \in R^{min}$, its new proportion α_r is recomputed at the end of an observation period and is given by $\alpha_r = n_r/n_{total}$, where $n_{total} = \sum_{r\in R} n_r$ is the total number of flows routed during an observation period. Recall that for a minhop path $r \in R^{min}$, $\gamma_r = \hat{\gamma}$. Hence $\alpha_r b_r = \frac{n_r}{n_{total}}\frac{\eta\gamma_r}{n_r} = \frac{n_r}{n_{total}}\frac{\eta\hat{\gamma}}{n_r} = \frac{\eta\hat{\gamma}}{n_{total}}$. This shows that the above method of assigning flow proportions for the minhop paths equalizes their flow blocking rates.

As in the *vcr* scheme, we use the minimum blocking probability among the minhop paths, $b^* = min_{r\in R^{min}} b_r$, as the reference to control flow proportions for the alternative paths. This is done implicitly by dynamically adjusting the maximum permissible flow blocking parameter $\gamma_{r'}$ for each alternative path $r' \in R^{alt}$. At the end of an observation period, let $b_{r'} = \frac{\eta\gamma_{r'}}{n_{r'}}$ be the observed flow blocking probability for an alternative path r'. If $b_{r'} > b^*$, $\gamma_{r'} := \max\{\gamma_{r'} - 1, 1\}$. If $b_{r'} < \psi b^*$, $\gamma_{r'} := \min\{\gamma_{r'} + 1, \hat{\gamma}\}$. If $\psi b^* \le b_{r'} \le b^*$, $\gamma_{r'}$ is not changed. By having $\gamma_{r'} \ge 1$, we ensure that some flows are occasionally routed along alternative path r' to probe its "quality", whereas by keeping $\gamma_{r'}$ always below $\hat{\gamma}$, we guarantee that minhop paths are always preferred to alternative paths in routing flows. The new proportion for each alternative path r' is again given by $\alpha_{r'} = n_{r'}/n_{total}$. Note that since $\gamma_{r'}$ is adjusted for the next observation period, the *actual* number of flows routed along alternative path r' will be also adjusted accordingly.

The *psr* scheme preserves the self-adaptivity of the theoretical *vcr* scheme by controlling the number of flows routed along a path r in each cycle using γ_r and by re-adjusting flow proportions after every observation period. For example, if the load along a path r increases, causing the number of flows blocked quickly reach γ_r, the source will automatically back off from this path by eliminating it from the eligible path set for the rest of the cycle. If this situation persists, at the end of the observation period, the new flow proportion for path r will be reduced. Likewise, if the load on path r decreases, its new flow proportion will be increased at the end of the observation period. This is particularly true for alternative paths with its dynamically adjusted γ_r. Furthermore, because the length of each cycle is not fixed but determined by how fast each eligible path reaches its maximal permissible blocks, the length of an observation period also varies. This self-adjusting observation period allows the *psr* scheme to respond to the system load fluctuations in an elastic manner. If the system load changes suddenly, the old flow proportions would result in rapid termination of cycles, which would in turn lead to faster ending of the current observation period. New flow proportions will thus be re-computed to adapt to the system load. On the other hand, if the system load is stable, the observation periods will also be stabilized, with increasingly accurate calibration of the flow proportions. As a result, flow proportioning will eventually converge to the equilibrium state.

Table II compares the simulation results obtained using the *psr* scheme with the corresponding numerical results obtained using the theoretical *vcr* scheme under various settings. The capacities of all bottleneck links are set to 20. The values for configurable parameters in *psr* were set to $\eta = 3$, $\hat{\gamma} = 5$, and $\psi = 0.8$. The table shows the proportions assigned to each path and also the overall blocking for each setting. In all the settings, the difference in proportions and the overall blocking between these two schemes is almost negligible. An interesting case is that of *duck* topology where *vcr* assigns zero proportion to the alternative path $1 \to 7 \to 8 \to 4 \to 9$ since it shares a bottleneck link $(4 \to 9)$ with minhop paths. The *psr* scheme routes 0.021 proportion of flows to this path. This is because *psr* has to route some flows to a path to probe its quality. However note that this is quite an insignificant proportion and doesn't severely affect the performance. These results show that the *psr* scheme closely approximates the *vcr* scheme.

It is interesting to contrast *psr* with some of the dynamic routing schemes proposed in the context of telephone networks. For example, the *dynamic alternative routing* (dar) scheme [4], [9] employs the *sticky routing* principle to re-route calls along alternative two-link paths, when the direct link is not available.



(a) impact of update interval ($\rho = 0.60$) (b) varying uniform load (c) varying hot load

Fig. 9. Performance comparison of *wsp* and *psr*

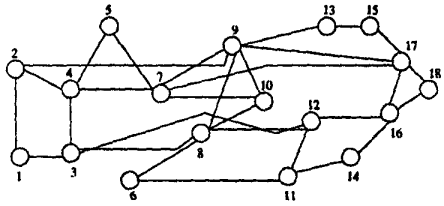


Fig. 8. The *isp* topology used in performance evaluation

Since in our setting we do not consider re-routing, a modified version of *dar* (with a dummy direct link), referred to as *sticky random routing* (srr), is used for comparison. The *srr* scheme routes flows through the same path till a connection setup results in a failure. Then a new path is chosen randomly from a set of feasible paths excluding the current path. Table III shows the overall blocking in *psr* and *srr* under various settings. The capacities of all bottleneck links are set to 20 except in one case where $c_2 = 10$ and $c_3 = 5$. Clearly *psr* outperforms *srr*[3]. This illustrates the advantage of proportional routing. Essentially *psr* does proportional routing while obtaining proportions through a form of sticky routing.

IV. Performance Evaluation and Analysis

This section evaluates the performance of the proposed localized QoS routing scheme *psr* in comparison with the global QoS routing scheme *widest shortest path* (wsp). We start with the description of the simulation environment and then compare the performance of *psr* and *wsp* in terms of the overall blocking probability, routing stability and overhead.

A. Simulation Environment

Figure 8 shows the *isp*[4] topology of an ISP backbone network used in our study (also used in [1], [10]). For simplicity, all the links are assumed to be bidirectional and of the same capacity, with C units of bandwidth in each direction. Flows arriving into the network are assumed to require one unit of bandwidth. Hence each link can accommodate at most C flows simultaneously. The flow dynamics of the network are modeled as follows (similar to the model used in [16]). Flows arrive at a source node according to a Poisson process with rate λ. The destination node of a flow is chosen randomly from the set of all nodes except the source node. The holding time of a flow is exponentially distributed with mean $1/\mu$. Following [16], the offered network load is given by $\rho = \lambda N \bar{h}/\mu LC$, where N is the number of source nodes, L the number of links, and $\bar{h}$ is the mean number of hops per flow, averaged across all source-destination pairs. The parameters used in simulation are $C = 20$, $N = 18$, $L = 60$, $\bar{h} = 2.36$, $\mu = 60$ sec. The average arrival rate at a source node λ is set depending upon the desired load.

B. Blocking Probability

Figure 9 compares the performance of *wsp* and *psr*. The values for configurable parameters in *psr* were set to $\eta = 3$, $\hat{\gamma} = 5$, and $\psi = 0.8$. It also shows the performance of naive random routing scheme *uniform random path* (urp) for reference. The *urp* scheme randomly selects a path from a set of pre-computed paths *without* using any QoS state information. The performance is measured in terms of the overall flow blocking probability, which is defined as the ratio of the total number of blocks to the total number of flow arrivals. In Figure 9(a), the overall blocking probability is plotted as a function of the update interval for an offered load of 0.60. Note that update interval is used only in *wsp*[5]. From this figure, we see that as the update interval of *wsp* increases, the blocking probability of *wsp* rapidly approaches that of *psr* and performs worse for larger update intervals. Figure 9(b) shows the blocking performance of these schemes as a function of the offered network load. The network load is varied from 0.50 to 0.70. The performance of *wsp* is plotted for three update intervals of 30, 60 and 120. It is clear that the blocking performance of *psr* is quite similar to that of *wsp* with update interval of 60 secs. This shows that *psr* using only local information performs comparably to global information exchange based QoS routing scheme such as *wsp* even when the update interval is reasonably small.

It is likely that a source node receives a larger number of

[3]For a more detailed comparison with *srr* and other similar schemes [9], [11], please refer to [13]

[4]Simulations were also carried out with other topologies and under different traffic conditions. The results were found to be similar [13] and not included here due to space limitations.

[5]We experimented with threshold based triggering also for updates. But with a clamp-down timer value T, the blocking performance found to be no better than simple periodic updates with an interval of T

flows to a few specific destinations [2], i.e, a few destinations are "hot". Ideally a source would like to have more up-to-date view of the QoS state of the links along the paths to these "hot" destinations. In the case of *wsp*, this requires more frequent QoS state updates, resulting in increased overhead. But in the case of *psr*, because of its adaptivity and statistics collection mechanism, a source does have more accurate information about the frequently used routes and thus alleviates the effect of "hot spots". We illustrate this by introducing increased levels of traffic between certain pairs of network nodes ("hot pairs"), as was done in [1]. Apart from the normal load that is distributed between all source-destination pairs, an additional load (hot load) is distributed among all the hot pair nodes. The hot pairs chosen are $(2, 16)$, $(3, 17)$, and $(6, 15)$. The normal load is fixed at 0.50 and the hot load is varied. Figure 9(c) shows the overall flow blocking probability as a function of the hot load. When the hot load is less than 0.01 the blocking performance of *psr* is comparable to *wsp* with update interval of 60 secs. But as the traffic between hot pairs increases, *psr* progressively does better in comparison to *wsp*. Particularly when the hot load is 0.06, the blocking performance of *psr* is same as that of *wsp* with an update interval of 30 secs. This not only shows the limitation of global QoS routing schemes such as *wsp* but also illustrates the advantage of self-adaptivity in localized QoS routing schemes such as *psr*.

C. Heterogeneous Traffic

The discussion so far is focussed on the case where the traffic is homogeneous, i.e., all flows request for one unit of bandwidth and their holding times are derived from the same exponential distribution with a fixed mean value. Here we study the applicability of *psr* in routing heterogeneous traffic where flows could request for varying bandwidths with their holding times derived from different distributions. We demonstrate that *psr* is insensitive to the duration of individual flows and hence we do not need to differentiate flows based on their holding times. We also show that when the link capacities are considerably higher than the average bandwidth request of flows, it may not be necessary to treat them differently and hence *psr* can be used *as is* to route heterogeneous traffic.

Consider the case of traffic with k types of flows, each flow of type i having a mean holding time $1/\mu_i$ and requesting bandwidth B_i. Let ρ_i be the offered load on the network due to flows of type i, where the total offered load, $\rho = \sum_{i=1}^{k} \rho_i$. The fraction of total traffic that is of type i, $\phi_i = \rho_i/\rho$. The arrival rate of type i flows at a source node, λ_i is given by $\lambda_i = \rho_i \mu_i LC/N\bar{h}B_i$, which is an extension of the formula presented in Section IV-A. To account for the heterogeneity of traffic, bandwidth blocking ratio is used as the performance metric for comparing different routing schemes. The bandwidth blocking ratio is defined as the ratio of the bandwidth usage corresponding to blocked flows and the total bandwidth usage of all the offered traffic. Suppose b_i is the observed blocking probability for flows of type i, then the bandwidth blocking ratio is given by $\frac{\sum_{i=1}^{k} \frac{b_i \lambda_i B_i}{\mu_i}}{\sum_{i=1}^{k} \frac{\lambda_i B_i}{\mu_i}}$. In the following, we compare the performance of *psr* and *wsp*, measured in terms of bandwidth blocking ratio, under different traffic conditions, varying the fractions ϕ_i to control the traffic mix.

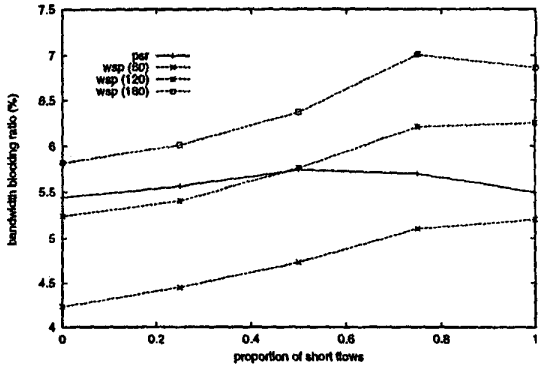


Fig. 10. Performance for flows with variable holding times

First, consider the case of traffic with 2 types of flows, each with different mean holding times but request for the same amount of bandwidth. Figure 10 shows the performance of *psr* and *wsp* for this case where $1/\mu_1$ and $1/\mu_2$ are 60 and 120 seconds respectively and $B_1 = B_2 = 1$. The load ρ is set to 0.60. The bandwidth blocking ratio is plotted as a function of the fraction ϕ_1 corresponding to type 1 flows (*short* flows). It is quite evident that the performance of *wsp* degrades as the proportion of *short* flows increases while that of *psr* stays almost constant. The behavior of *wsp* is as expected since the shorter flows cause more fluctuation in the network QoS state and the information at a source node becomes more inaccurate as the QoS state update interval gets larger relative to flow dynamics. On the contrary, *psr* is insensitive to the duration of flows, which can be explained as follows.

The behavior of *psr* is not surprising since Erlang formula is known to be applicable even when the flow holding times are not exponentially distributed and blocking probability depends only on the load, i.e., the ratio of arrival rate and service rate. For the above case of two types of flows, the aggregate arrival rate, λ, is given by $\lambda = \lambda_1 + \lambda_2$ and the mean holding time, $1/\mu$, is given by $\frac{1}{\mu} = \frac{1}{\mu_1}\frac{\lambda_1}{\lambda_1+\lambda_2} + \frac{1}{\mu_2}\frac{\lambda_2}{\lambda_1+\lambda_2}$. This heterogeneous traffic can then be treated as equivalent to homogeneous traffic with arrival rate λ, mean holding time $1/\mu$ and the corresponding load $\lambda/\mu = \lambda_1/\mu_1 + \lambda_2/\mu_2$. So for a given load, the blocking probability would be same irrespective of the mean holding times of individual flows. That is why the performance of the theoretical scheme, *vcr* depends only on the overall offered load and not on the types of traffic. The practical scheme, *psr* inherits this property from *vcr* and hence *psr* can be employed *as is* to route flows with mixed holding times.

Now, consider the case of traffic with 2 types of flows, each requesting for different amount of bandwidth but having same mean holding time. As above, the load ρ is set to 0.60. Figure 11 shows the performance of *psr* and *wsp* for this case where $B_1 = 1$, $B_2 = 2$, and $1/\mu_1 = 1/\mu_2 = 60$ sec. Once again, the bandwidth blocking ratio is plotted as a function of the fraction ϕ_1 corresponding to type 1 flows (*small* flows). As expected, both schemes perform better as the proportion of *small* flows increases. It can also be seen that there is no discernible change in the relative performance of *psr* with respect to *wsp*. Our re-

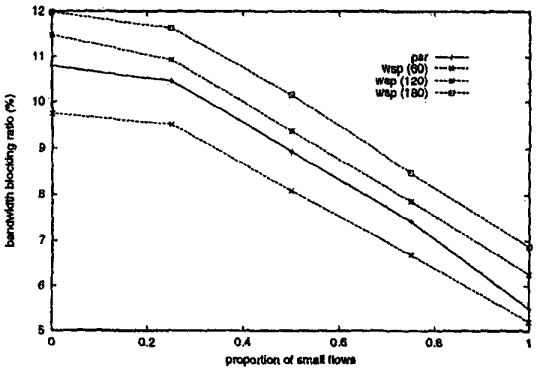


Fig. 11. Performance for flows with variable bandwidth requests

sults show that when the requested bandwidth is significantly smaller than the link capacity, it may not be necessary for *psr* to differentiate between different bandwidth requests. However, when bandwidth requests are relatively large it is possible that a path that is good for one bandwidth request may not be even feasible for another bandwidth request. Furthermore, Erlang formula cannot be applied directly to this case of traffic with variable bandwidth requests. In addition, since the amount of bandwidth requested by a flow is known at the time of path selection, it makes sense to utilize this knowledge in categorizing them into bandwidth classes and routing them accordingly. We are currently exploring class based routing approaches using multi-service loss models [14] for routing flows with variable bandwidth requests.

D. Routing Stability

An essential feature of a good routing scheme is its ability to avoid routing oscillations and thus ensure stability. It was shown [16] that out-of-date information due to larger update intervals can cause route flapping in schemes such as *wsp*. On the other hand, the *psr* scheme doesn't exhibit such route flapping behavior. The fluctuation in link utilization is much less and more gradual. There are two fundamental reasons for this stability of *psr*. First, in *psr* each source performs routing based on its own *local view* of the network state. Routing based on such a "customized view" avoids the undesirable *synchronized mass reaction* that is inherent in QoS routing scheme based on a global view. Second, *psr* does proportional routing with a proportion assigned to a path reflecting its quality. A relatively better path is favored by sending larger proportion of traffic to it. It doesn't pick just one "best" path. Thus a localized proportional routing scheme such as *psr* is intrinsically more stable than global QoS routing schemes such as *wsp* which select the "best path".

E. Routing Overhead

We now compare the amount of overhead involved in these two routing schemes. Path selection in *wsp*, using a variant of Dijkstra's algorithm, takes $O(N^2)$ time where N is the number of nodes in the network. Assuming pre-computation of a set of paths R to each destination, it still need to traverse $O(L)$ links, where L is the total number of links in the set R. On the other hand, the path selection in *psr* is simply an invocation of *wrrps* whose worst case complexity is $O(|R|)$. Now look at the overhead involved in collecting the information required for path selection. In *wsp*, every router is responsible for maintaining consistent QoS state and generating updates about all the links adjacent to it. This incurs both communication and processing overhead. In contrast, in *psr*, only edge routers need to keep track of route level statistics and recompute proportions after every observation period. The bookkeeping for maintaining statistics involves only increment and decrement operations costing only constant time per flow. The proportion computation procedure in *psr* itself is extremely simple and costs no more than $O(|R|)$.

V. Conclusions

This paper focused on *localized* QoS routing schemes where the edge routers make routing decisions using only "local" information and thus reducing the overhead at core routers. We first described *virtual capacity based routing* (vcr), a theoretical scheme based on the notion of *virtual capacity* of a route. We then proposed *proportional sticky routing* (psr), an easily realizable approximation of *vcr* and analyzed its performance. We demonstrated through extensive simulations that *psr* scheme is indeed simple, stable, and adaptive. We conclude that the *psr* scheme, with low overhead and comparable performance, is a better alternative to global QoS routing schemes.

References

[1] G. Apostolopoulos, R. Guerin, S. Kamat, S. Tripathi, "Quality of Service Based Routing: A Performance Perspective", ACM SIGCOMM 1998.
[2] J. Chen, P. Druschel, D. Subramanian, "A New Approach to Routing with Dynamic Metrics", IEEE INFOCOM 1999.
[3] E. Crawley, R. Nair, B. Rajagopalan, H. Sandick, "A Framework for QoS-Based Routing in the Internet", *Work in Progress*, Internet Draft, July 1998.
[4] R.J. Gibbens, F.P. Kelly, P.B. Key, "Dynamic Alternative Routing: Modelling and Behaviour", Teletraffic Science, pp. 1019-1025, Elsevier, Amsterdam, 1989.
[5] R. Guerin, S. Kamat, A. Orda, T. Przygienda, D. Williams, "QoS Routing Mechanisms and OSPF Extensions", *Work in Progress*, Internet Draft, March 1997.
[6] R. Guerin, A. Orda, "QoS-Based Routing in Networks with Inaccurate Information: Theory and Algorithms", IEEE INFOCOM 1997.
[7] F.P. Kelly, "Routing in Circuit-Switched Networks: Optimization, Shadow Prices and Decentralization", Advances in Applied Probability 20, 112-144, 1988.
[8] F.P. Kelly, "Routing and capacity Allocation in Networks with Trunk Reservation", Mathematics of Operations Research, 15:771-793, 1990.
[9] P.B. Key, G.A. Cope, "Distributed Dynamic Routing Schemes", IEEE Communications Magazine, pp. 54-64, Oct 1990.
[10] Q. Ma, P. Steenkiste, "On Path Selection for Traffic with Bandwidth Guarantees", IEEE ICNP 1997.
[11] K.S. Narendra, P. Mars, "The Use of Learning Algorithms in Telephone Traffic Routing - A Methodlogy", Automatica, vol. 19, no. 5, pp. 495-502, 1983.
[12] S. Nelakuditi, R.P. Tsang, Z. Zhang, "Quality-of-Service Routing without Global Information Exchange", IWQOS 1999.
[13] S. Nelakuditi, Z.-L. Zhang, R.P. Tsang, "Adaptive Proportional Routing: A Localized QoS Routing Approach", Technical Report, Department of Computer Science, University of Minnesota, July 1999.
[14] K.W. Ross, "Multiservice Loss Models for Broadband Telecommunication Networks", Springer-Verlag, 1995.
[15] A. Shaikh, J. Rexford, K. Shin, "Evaluating the Overheads of Source-Directed Quality-of-Service Routing", ICNP 1998.
[16] A. Shaikh, J. Rexford, K. Shin, "Load-Sensitive Routing of Long-Lived IP Flows", ACM SIGCOMM 1999.
[17] Z. Wang, J. Crowcroft, "Quality-of-Service Routing for Supporting Multimedia Applications", IEEE JSAC, Sept 1996.
[18] Z. Zhang, C. Sanchez, B. Salkewicz, E. Crawley, "Quality of Service Extensions to OSPF", *Work in Progress*, Internet Draft, September 1997.