

QoS Routing Computation with Path Caching: A Framework and Network Performance

Deep Medhi*
Computer Networking
School of Interdisciplinary Computing & Engineering
University of Missouri–Kansas City
Kansas City, MO 64110
Email: dmedhi@umkc.edu

July 2001, revised March 2002

Abstract

In this paper, we present a framework for QoS routing computation with path caching. The framework has three phases to allow different levels of information to be processed at different time scales towards effectively meeting QoS requirement of a newly arrived flow. Path caching is introduced in the first phase to allow for selection and filtering in subsequent phases. We describe several routing schemes which can fit into this framework.

Through simulation results, we show where and how the benefit of path caching can be exploited depending on the number of paths cached, and where to invoke other controls. Our results show that QoS routing by itself can not improve network and/or service performance unless controls such as trunk reservation and source-based admission control are also activated. The invocation of these functions can also allow maximum benefit out of the path caching framework.

I Introduction

Quality-of-Service (QoS) routing has been receiving significant attention in the past few years with the emergence of service offerings that require quality-of-service guarantees on the Internet. QoS guarantee refers to meeting service requirements such as bandwidth guarantee, packet loss rate, and/or jitter delay for a request. On the other hand, typically, these types of requirements aren't required to be satisfied by best-effort services.

Currently, the primary routing decision for best-effort services on the Internet is at the packet level, and for this, shortest-path based routing is commonly deployed. In most cases, best-effort services operate in a destination-oriented routing mode. What this means is that the source of the packet isn't taken into account when the routing decision is made at any router; instead, the destination is taken into account on a hop-by-hop basis. Further, currently, most routing protocols (such as OSPF) are deployed with static link metric (such as the interface cost being related to the speed of the link) to determine the shortest path [14].

While the destination-oriented, shortest-path based routing paradigm has worked well for current best-effort services, a new or different mode of operations is needed for services that require QoS guarantee [4]. First, a source-destination based routing mode is more desirable than purely destination-based routing. With this, links that are congested can be avoided with routing decision being made at the source (based on network information available to routers). Second, the use of link metric such as the static link-speed-based interface cost is also problematic since services that require quality-of-service guarantees can not effectively use this information to

*Supported by DARPA and Air Force Research Lab, Air Force Materiel Command, USAF, agreement No. F30602-97-1-0257, University of Missouri System Research Board grant K-3-40605 and NSF grant NCR-9506652.

select a good path from the source to the destination. Finally, given the source-destination based routing paradigm, the ability to provide a set of possible paths that meet QoS guarantee than just the shortest-path is also desirable.

Before we go further, we briefly review the QoS requirement issue. Typically, services that need QoS guarantee operate at the flow level and require QoS guarantee for the duration of the flow. A *flow* is referred to as an IP packet stream from a source to a destination with an associated QoS [4]¹ and has a finite duration. While it is not necessary that the same path be used for the entire duration of the flow, we will assume that the path stays the same for the entire duration – this is often referred to as *route pinning*. When a flow request arrives, there is a flow set-up phase that can typically perform functions such as route determination, signalling along the path to the destination and QoS checking before the flow is set up; in essence, a flow request faces a flow set-up time before it can be connected. For the services that require QoS guarantee, the flow set-up phase needs to ensure that the QoS guarantee can be provided for the entire duration of the flow; otherwise, the flow request is denied. It is possible that at the user level, there is a retry attempt for a denied flow request. An important point to observe is that the route selection decision is at the flow-level granularity rather than at the packet level. Packet forwarding function is only activated at the packet level along the path, already established during the flow set-up phase, and for the entire duration of the flow due to route pinning.

In this paper, we present a framework for QoS routing computation which is motivated by several goals:

- We want to reduce the impact on the flow set-up time at least from the routing computation perspective. That is, an important goal is to minimize the flow-set up time.
- We want to avoid user-level retry attempts; i.e., it is preferable to do retry *internal* to the network as long as the flow set-up time isn't drastically affected. It is important to note that user-level retry attempts can not be completely avoided, at least in a heavily-loaded network (i.e., a network where the ratio of traffic to network bandwidth is at a level beyond the normally acceptable tolerance for service guarantee).
- To be able to do internal-to-the network retry, the network needs to have the capability to select a route from a number of possible paths very quickly, and also possibly have the *crankback* capability. Crankback[4] refers to the network functionality that allows a flow, which is in the process of being set-up along a path, to backtrack to the source node to try along another path.
- Sometimes a flow request attempting the very first outgoing link (for a path identified by the routing process) may not be able to access the link (this will be clear when we discuss the routing schemes); this is referred to as *source-link-overflow*. If this is the case, it should be able to try a second path as dictated by the routing process for overflowing the request.
- The framework should allow the ability to incorporate several routing schemes so that network providers can choose the appropriate one depending on their performance goal.

The source-link-overflow idea along with the desire to avoid user-level retry and to allow crankback suggests that having multiple paths choices can be beneficial in a QoS routing environment; this is often referred to as

¹For consistency, we will try to follow the terminology used in RFC2386 [4].

the *alternate path routing* concept. Taking the above goals into account, we describe a QoS routing computation framework with path caching. Here, *path caching* refers to the ability to store multiple possible paths *ahead of time* between a source and a destination so that a flow request can use any of these paths based on a specific set of rules.

Our discussion in this paper centers around QoS routing computation within an autonomous system, i.e., in an intra-domain routing environment. A flow request with QoS parameters can arrive at a router internal to the autonomous system, or at a border router from another autonomous system; similarly, the flow may be destined for a router internal to the network, or to a border router for a destination outside the autonomous system. For the purpose of our discussion here, we do not distinguish between an internal router or a border router. Instead, we will refer to the router where a flow enters the network as the source or ingress node (router), and the router where the flow leaves the network as the destination or egress node (router). We assume that a router has high-level admission control capability used to determine whether to admit a flow to the network. While a part of this functionality is used for policy decision, we will restrict our discussion to the admitability of the flow either due to QoS requirement or due to any constraint imposed by the network, not due to any policy reason. Thus, we will refer to it as *source-based (flow) admission control*. Also, in this work, we will limit the QoS requirements to hop count or interface cost-based path requirement, and services that require bandwidth guarantee. Hop count or interface-speed derived cost has additive cost property while bandwidth requirement has non-additive cost property when it comes to determining routes. Note that the limit on hop count is not necessarily imposed by the services directly; it can be imposed by the network providers to impose better QoS to any specific service. We will discuss at the end how other QoS requirements can be incorporated.

In the rest of the paper, we will describe the framework in more details, and show how several routing schemes may be incorporated within this framework. We'll then present some computational results to understand the effect of path caching as well as the implication of different routing schemes.

II Routing Computation Framework

The basic idea behind this framework addresses the following: how is the selection of paths done, when are they selected, and how are they used by newly arrived flows. For flows with bandwidth guarantees, another important component that can complicate the matter is the definition of the cost of a path based on possibly both additive and non-additive properties. Later, we will consider our framework using an extended link-state protocol concept. Before we do that, we describe the three phases of our framework: (1) Preliminary Path Caching (PPC) phase, (2) Updated Path Ordering (UPO) phase, and (3) Actual Route Selection (ARS). Each of these phases operates at different time scales.

The first phase, PPC, does preliminary determination of a set of possible paths from a source to destination node, and their storage (caching). A simple case for this phase is to determine this set at the time of major topological changes. PPC, in the simplest form, can be thought of as topology dependent, i.e., if there is a change in the major topological connectivity, then the PPC phase may be invoked. On the other hand, it can be somewhat intelligent, i.e., if a link availability is expected to be less than a certain threshold for a prolonged duration or if the link is scheduled for some maintenance work, then PPC can also be used for pruning the link, and determining a new set of cached paths. Essentially, PPC uses a coarse-grain view of the network, and determines a set of

candidate paths to be cached. A simple mechanism to determine the set of paths for each source node to each destination node may be based on hop-count or interface cost or some administrative weight as the cost metric using the K -shortest paths algorithm [8]. Thus, for this phase, we assume the cost-metric to be additive.

The second phase, UPO, narrows the number of QoS acceptable paths; this module uses the most recent status of all links as available to each source node. Since, the PPC phase has already cached a set of possible paths, this operation is more of a compare or filter to provide a set of QoS acceptable paths. Further, for a specific service type or class, this phase may also *order* the routes from most acceptable to least acceptable (e.g. based on path residual bandwidth), and will, in general, have a subset of the routes "active" from the list obtained from the PPC phase. In this phase, the cost-metric can be both additive (e.g., delay requirement), or non-additive (bandwidth requirement), although we'll only use non-additive bandwidth requirement as cost metric in our examples. Another important factor to note about the UPO phase is that the value of the update interval may vary, even for the same routing scheme; for simplicity, we'll refer to this as the routing update interval (RUI).

The third phase is ARS. From the UPO phase, we already have a reasonably good set of paths. The ARS phase selects a specific route on which to attempt a newly arrived flow. The exact rule for selecting the route is dependent on a specific route selection procedure. The main goal in this phase is to select the actual route as quickly as possible based on the pruned available paths from the UPO phase.

There are several advantages of the three phase framework:

- Several different routing schemes can be cast in this framework (discussed in the next section).
- It avoids on-demand routing computation (from scratch); this reduces impact on the flow set-up time significantly since paths are readily available; there is no "cost" incurred from needing to compute routes from scratch *after* a new flow arrives.
- The framework can be implemented using a link-state routing protocol (with some extension). For the PPC phase, some topology information, for example, are needed to be exchanged at coarse-grain time windows. During the UPO phase, periodic update on status of link usage is needed at a finer grain time window. Since different information about links are needed at different time granularity for use by the PPC and the UPO phase, we refer to this as the *extended* link-state protocol concept.
- Due to the use of the extended link-state protocol, the entire environment can work in a distributed manner (without requiring centralized computation).
- Further, each of the three phases can be independently modified without affecting the other ones. For example, in the PPC phase, the K shortest paths can be computed either based on pure hop count or other costs such as link-speed based interface cost. In some schemes the UPO phase may not be necessary.

A possible drawback of the framework is that path caching will typically require more memory at the routers to store multiple paths; this will certainly depend also on how many paths are stored. On the other hand, with the drop in memory prices, a path caching concept is more viable than ever before. Additionally, there is some computational overhead due to K -shortest path computation on a coarse-scale time window. Our experience has been that K -shortest path computation takes only a few seconds to generate five to ten paths in a fifty-node network on a off-the-self computer. Thus, this overhead isn't remarkable since it's done in the PPC phase. If needed, a router architecture can be designed to include a separate processor to do this type of computational work periodically.

III Routing Schemes

In this section, we describe four routing schemes, and their relation to the three-phase framework.

Maximum Available Capacity Routing with Periodic update and Crankback (MACRPC):

In the PPC phase, a set of paths is generated and cached based on, say, the hop-count limit using the K -shortest path algorithm. If there is a direct link-path between any ingress-egress node, then this path is always cached as the first path (in a fully connected network, every node pair has a direct link path). For the UPO phase, the most recent update received on the status of various links obtained through the link-state protocol is used to periodically determine the cached paths in the order of maximum available (residual) to least available capacity. Any path that does not have bandwidth available is temporarily marked as an unavailable path. The duration of the update period (*UPO Periodicity Interval*) is a variable parameter of the scheme. In this phase, with the pruned path set, the currently available capacity can also be stored. This phase uses the non-additive cost property of bandwidth for ordering the paths (see Appendix).

When a new flow arrives, the ARS phase is activated. In this phase, the direct-link path is always tried first (if this is available or exists). If the direct-link path does not exist, the MACRPC scheme first attempts the route with the maximum available capacity as determined during the UPO phase. Depending on the update period for the UPO phase, the reduced routing set from which a path of maximum available capacity is selected for a newly arriving flow can be out-dated; in other words, the path with the most available bandwidth from the last UPO phase may not have any bandwidth available to accommodate this flow, at least on some link(s) along the paths. This can occur since link information from the last update can be used by other source-destination node pairs to determine their own ordered paths which can grab bandwidth if a flow request arrives for such a pair *before* this pair.

Thus, due to inaccuracy of information in between the interval of updates, it is possible that a newly-arrived flow is blocked on this path by the link-admission control procedure (on a link that does not have any bandwidth available at that instant). The MACRPC scheme allows the flow to crankback and try the next best path from the current path set, and keeps trying until all the possible paths which have been cached at the time of last update interval have been exhausted (unless there is a limit imposed on the number of cranked paths to be attempts to limit the flow-set-up time). The flow is considered blocked *only* after all these paths are tried. Note that this scheme also has source-link-overflow which is subsumed by the crankback procedure. The performance of this scheme is expected to be dependent on the routing update interval which we will discuss in the results section by considering different update periods.

Maximum Available Capacity Routing with Periodic update and NO Crankback (MACRPNC):

This scheme is very similar to MACRPC and follows essentially the same three phases. The only difference is that in this scheme, crankback is completely turned off; however, the source-link-overflow option remains activated.

Cached Sticky Random Adaptive Routing (CaSRAR):

This routing scheme is an extension of the sticky random routing scheme, Dynamic Alternate Routing (DAR) [5] originally proposed for circuit-switched voice networks. In the PPC phase, a set of paths is cached like the other schemes. On the other hand, in this scheme the UPO phase is not necessary. During the ARC phase, a

routing table maintains two paths: the direct path (if one exists), and an alternate path. A newly arrived flow tries the direct path first, if it exists, and the bandwidth can be guaranteed. Otherwise the flow tries the stored alternate route and leaves the system (i.e., is blocked) if bandwidth is not available on this path. In case a newly arrived flow is blocked and cleared, the alternate path is updated by picking randomly a new path from among the cached paths. This is then used by future flow arrivals; on the other hand, if the flow is accepted on the alternate path, the stored path is *not* changed ("sticky"). Note that the link state protocol is invoked only for the PPC phase; beyond that no other exchanges are necessary. There is no crankback used in this scheme. While the UPO phase is not necessary, this can be activated to prune the set of cached paths, if needed.

In addition to the above schemes, we also consider the following "reference" routing scheme:

Maximum Available Capacity Routing with Instantaneous Computation (MACRIC):

This scheme has the same PPC phase. In this routing scheme, when a new flow arrives, the entire network is scouted to find the path with the most available capacity; this process is invoked for *every* newly arrived flow. MACRIC can be thought of as the limiting case of MACRPC and MACRPNC. MACRIC is an utopian routing scheme with completely on-demand computation based on instantaneous availability of the link state information. It is hard to implement in practice without significantly increasing load on the network due to link-state exchanges; and therefore, it is considered here primarily for benchmarking. Note that crankback is not necessary in this scheme since the paths are sorted based on instantaneous knowledge of available bandwidth for all links; thus, if the first path in the ordered set doesn't have bandwidth, there is no point attempting the other paths. In the context of the framework, we can think of the UPO phase as being merged with the ARS phase.

A Remark

A fundamental component of the QoS routing framework is path caching. With this, in the PPC phase, a K-shortest path algorithm is used to generate a set of paths which are cached. At this phase, the cost-metric used is additive. For the routing schemes, an extended link-state protocol is used for disseminating the status of the link (different information) at the PPC phase and the UPO phase. Since paths are already cached, the UPO phase can use a simple filtering mechanism to order paths based on available bandwidth (for services that require bandwidth guarantee for QoS). If there are services that have other QoS requirements such as path delay, these requirements can be easily incorporated in the UPO phase as additional filters.

Recall that an important goal of reducing the impact on flow set-up time is addressed by the framework through the notion of path caching. Due to the three-phase framework, the newly arrived flow attempts one of the paths already pruned by the UPO phase – so there is no on-demand route computation delay in this phase. Depending on the periodicity of the UPO phase and the arrival of the link-state advertisement, the pruned path set can have outdated information. Thus, some newly arrived flows can be assigned to a path that may not have any available bandwidth at this instant. This can not be completely avoided unless the frequency of the update interval is reduced; if this is done, then more frequent link state advertisement would be necessary which leads to an increase in network traffic.

A critical issue that faces flow-based routing is the scalability of maintaining per-flow states in the core routers. In our framework, this is not necessary. The maintenance of per flow state information can be pushed to the edge routers; the core routers maintain just the link state information which is disseminated to others.

IV Results and Observations

So far, we have presented the QoS routing computation framework and examples of routing schemes that can be cast in this framework. Besides the ability to incorporate different routing schemes, we have already discussed the overall benefit of the QoS routing computation framework in terms of its capability to help reduce flow set-up time, for example, by doing some computational work ahead of time in the PPC phase. What remains to be discussed is the performance comparison of the routing schemes. The intent here is to show the performance difference and understand whether and when other control schemes may possibly be needed to maintain good network performance. Extensive simulation studies with the routing schemes (and other variations) for different networks (both for fully-connected and sparse networks, different network sizes, and with symmetric and asymmetric traffic) and for a number of network load conditions were conducted. For the sake of conciseness, we will highlight a few key observations (for detail results, see, for example, [10, 11, 12, 13]).

Before we go further, we briefly discuss the performance indicator used here to measure network performance. In a network with *heterogeneous* services (i.e. where different flows require different bandwidth guarantee), the overall network-wide performance criterion is best captured by the bandwidth denial ratio (BDR). Let \mathcal{N} be the set of flows that arrives to a network where the bandwidth requirement of flow j is given by w_j . Let \mathcal{N}_B be the set of flows which were denied service to the network. BDR is then given by

$$BDR = \frac{\sum_{j \in \mathcal{N}_B} w_j}{\sum_{j \in \mathcal{N}} w_j}.$$

If all flows have the same per flow bandwidth requirement (i.e., the *homogeneous* service case), then this indicator reduces to the flow blocking probability given by $\frac{\#(\mathcal{N}_B)}{\#(\mathcal{N})}$, where $\#(\cdot)$ denotes the cardinality of a set. When there are different service classes where the bandwidth requirement within each class is homogeneous, then it is important to consider both flow blocking in each class as well as the BDR for the network.

In a reservation-oriented network where a flow may be denied entry to the network due to non-availability of resources, an important parameter (which is not necessarily within the control of the routing scheme) can have noticeable impact on the network performance. This parameter is traditionally referred to as the *trunk reservation*² parameter; its necessity was first reported by Weber almost four decades ago [15] for circuit-switched networks. It is known that for an alternate-routing network without any trunk reservation, network bi-stability is possible [7]. Another important factor happens to be flow admission control scheme (at the source). We will touch on these factors as we discuss various network scenarios.

We first start with results on flow blocking for the homogeneous service case as the number of cached paths K changes from 2 to 15 (for a ten-node fully-connected network); this is reported in Figure 1 for both the case of no reservation and with trunk reservation (set at 40%³). It is interesting to note that, for the no reservation case, the increase of cached paths does not necessarily result in improvement in performance for *all* routing schemes. Only for MACRPNC, do we see improvement. On the other hand, with trunk reservation activated, performance can improve with the increase in K for *all* routing schemes. This substantiates the claim on performance degradation

²Trunk reservation saves a part of the link capacity for its own direct traffic in the sense that when this amount is all that is left (while the rest of the bandwidth has been allocated to existing flows), an alternate routed flow is not admitted to this link, but a flow for the direct traffic can still be admitted that can use this bandwidth.

³While a very high trunk reservation value may not be used in an operating network, the intent here is to show the differences.

in the absence of trunk reservation as reported in [7]. Further, our result shows that this behavior is not necessarily consistent for *all* routing schemes. For the utopian scheme, MACRIC, the performance degrades drastically as K increases when there is no trunk reservation. Although this may sound surprising, this is possibly caused due to over-use of multiple-links paths through instantaneous checking which leads to local optimization, and bi-stability. We observe the same problem with MACRPC when there is no trunk reservation. Overall, CaSRAR and MACRPNC are more robust in the absence of trunk reservation. However, in the presence of high trunk reservation, as K increases we found MACRIC and MACRPC to have better performance than CaSRAR and MACRPNC. Overall, these results show that path caching is indeed helpful; on the other hand, the actual routing schemes and factors such as trunk reservation do matter.

Next we discuss the case of heterogeneous services where three different service classes with differing bandwidth requirement for each service class are offered. We consider two cases: the network capacity in the first one is dimensioned⁴ for low BDR (less than 1%) while the second one is dimensioned for moderately high BDR (over 5%). From the scenario where the network is dimensioned for low BDR (Figure 2), we found that in the presence of trunk reservation, as K increases the BDR decreases for all schemes (similar to the homogeneous case). On the other hand, this is not true when the network is dimensioned for moderate BDR (Figure 3), even in the presence of moderate trunk reservation. The pattern is somewhat closer to the homogeneous case with no trunk reservation. What we can infer is that even in the presence of trunk reservation, the ability to hunt over multiple paths through crankback is beneficial in a network designed for low BDR, but crankback can be detrimental when the network is designed for moderately high BDR as it impacts network performance (and also can lead to higher flow set-up time due to frequent path hunting). Please see [13] for more details.

Now we discuss briefly the role of the UPO phase. Recall that different routing update interval (RUI) parameter values can be used for the UPO phase. As one would guess, with more frequent updates (i.e., for a smaller value of RUI), the inaccuracy in link-state information decreases. It is observed that both schemes MACRPC and MACRPNC give better performance with more frequent updates as one would intuitively guess. On the other hand, it appears that inaccuracy in link-state information can be well-compensated by the availability of crankback in a network designed for low BDR. Specifically, we note that MACRPC with RUI of 360 seconds has much lower BDR than MACRPNC with RUI of 120 seconds (Figure 2). However, the reverse relation holds when the load is moderately high (Figure 3). We also saw in an earlier example (through MACRIC) that instantaneous information update isn't always beneficial in terms of network performance (as well as negatively affecting flow set-up time considerably). Overall, we can infer that inaccuracy in link-state information isn't necessarily bad, and in fact, can be well-compensated through path caching; in any case, the specifics of the routing scheme do play a role here.

So far we have discussed performance using the network-wide indicator BDR. We are next interested in understanding the effect on each service class. For this, we have considered three service classes in increasing order of bandwidth requirement, i.e., the first service (s1) class has the lowest bandwidth requirement per flow, while the third service class(s3) has the highest bandwidth requirement per flow. For a network dimensioned for low BDR, we found that with moderate to large number of path caching, CaSRAR and MACRPNC tend to give poorer performance to the higher bandwidth service class (s3), whether the network is fully- or sparsely connected (Figure 2 shown here for the sparsely connected case). Further, the inaccuracy of routing information due to the update

⁴Dimensioning or sizing refers to determining the capacity needed in a network to carry a given traffic offered load at a pre-specified level of performance guarantee.

interval of the UPO phase does not seem to affect MACRPC for different service classes, but can noticeably affect MACRPNC (Figure 2). To check whether the same behavior holds, we increased the load uniformly for all service classes. We made a couple of interesting observations (Figure 3): the lowest bandwidth service (s1) has uniformly low flow blocking for *all* routing schemes; however, the highest bandwidth service class (s3) is worst affected under MACRPC at the expense of the lower bandwidth classes; i.e., MACRPC is more unfair to higher-bandwidth services as the network load uniformly increases. In general, we found CaSRAR to work better than the others schemes in providing smaller variation in performance differences seen by different service classes.

While it has been known that higher bandwidth, reservation-based services see worse performance than lower bandwidth, reservation-based services in a single-link system [6], these results indicate that this behavior holds as well in a network *with* dynamic routing and trunk reservation. In other words, routing and trunk reservation can *not* completely eliminate this unfairness. Thus, in a network, if fairness in terms of grade-of-service to different service classes is desirable, then additional mechanisms are needed. In this context, a concept called *service reservation* beyond traditional trunk reservation has been proposed [10]. This concept can be manifested, for example, through source-based admission control at the time of flow arrival. While a good source-based admission control scheme for a general topology network in the *presence* of QoS routing operating in a link-state protocol environment and trunk reservation remains a research problem, a probabilistic source-based admission control scheme for fully-connected networks in the presence of routing and for two services case has been presented in [10]. The ability to provide service fairness in terms of fair grade-of-service using this source based admission control scheme in the presence of routing and trunk reservation is shown in Figure 4. This is shown for three different values of network load with two service class scenario (shown for normal load “lf-1.0”, 5% s2 overload “lf-s2”, and 5% network-wide overload “lf-1.05”, all for MACRPC). The right-most entries (corresponding to $p1 = 1$) denote the *no* source-based admission control case. As we can see, with the increase in load, the higher bandwidth service suffers the most in the absence of source-based admission control. As the admission control parameter is adapted (by changing $p1$ towards 0.8) to invoke different levels of source-based admission control, it can be seen that service level fairness in terms of grade-of-service can be achieved.

Finally, we discuss network performance impact due to network traffic dynamics. To show this we consider a homogeneous service, fully-connected network where one source-destination node pair has dynamic traffic while the rest of traffic pairs has stationary traffic (no source-based admission control is included here). For our study, the dynamic traffic has been represented through a time-dependent, stationary process that follows a sinusoidal traffic pattern. For the case with no trunk reservation, we have found that MACRPC has much worse performance than both CaSRAR and MACRIC as traffic changes for the dynamic traffic class (pair); CaSRAR adapts very well with traffic changes although it has no UPO phase. It is interesting to note that just the presence of dynamic traffic between a source-destination node pair can cause the rest of the (stationary) traffic to show dynamic performance behavior (Figure 5). When trunk reservation is imposed, we noticed two things (Figure 6): (1) MACRPC performs better than CaSRAR for the dynamic traffic, and (2) the imposition of dynamic performance on the stationary traffic (from the dynamic traffic class) is no longer there. Also, we found that the overall performance improves in the presence of trunk reservation in a dynamic traffic scenario (similar to the stationary traffic case). From these results an important question, although not directly within the purview of routing, arises: should a network allow a dynamic traffic stream/class to impose its behavior on a stationary traffic stream? In other words, should a stationary traffic stream suffer higher flow blocking just because the load for the dynamic traffic stream is increasing?

To our knowledge, this can not be addressed alone through our three-phase QoS routing framework or any other QoS routing framework. On the other hand, the impact can be controlled through the use of trunk reservation. For additional results in the presence of dynamic traffic, (including heterogeneous services), see [12].

V Related Work

There has been tremendous amount of work in the QoS routing area. For the sake of space, a few work are discussed here.

In recent years, the concept of shortest-widest path has been proposed and discuss for QoS routing [9]. This concept is somewhat similar to MACRIC, but does not address the notion of path caching or inaccuracy due to periodic availability of link-state information. While we use the idea of K -shortest path paradigm for path caching because it allows us to use the notion of filtering very easily in our routing computation framework, others have addressed QoS routing from the perspective of Dijkstra-based or Bellman-Ford based shortest-path approach [2]. In another work, Guerin *et al* [1] have clearly articulated different performance perspectives for QoS routing.

We like to point out that the path caching idea by itself is nothing new. In fact, dynamic non-hierarchical routing (DNHR) which was initially deployed in AT&T's long-distance voice network in 1984 cached multiple sets of ordered paths [3]; each set was used for a particular time period during the day. Path ordering was done off line where at most two-link paths were allowed for each source-destination pair, and there was no concept of separate PPC and UPO phases as we proposed here; nor was link-state protocol used.

VI Summary

In this work, we have presented a fairly generic QoS routing computation framework based on path caching which can operate in the presence of a link-state protocol. An advantage of this framework is that various distributed routing schemes can fit into this framework.

The framework has three phases: PPC, UPO and ARS. This framework was motivated by our interest to reduce impact on the flow set-up time, yet use the link-state protocol to use different information effectively in two different time scales. Different routing schemes implemented in this framework have allowed us to see how performance objectives can be met for heterogeneous services.

Through computational results, we have observed that in a lowly loaded network, inaccuracy in link-state information due to periodic routing update in the UPO phase can be well-compensated for by having the crankback function, as we saw with MACRPC. It is known that in such situations crankback doesn't increase flow set-up time considerably. On the other hand, in a moderately loaded network, crankback can be detrimental unless trunk reservation is also present; further, too much crankbacking in a moderate to highly-loaded network can add to the flow set-up time also. When a number of paths are cached, having instantaneous information is not always beneficial; not only can it deteriorate network performance, it can also increase the flow set-up time. In other words, inaccuracy due to delayed link-state information isn't necessarily bad. We have found CaSRAR to be a remarkably good scheme. It works very well under most conditions, especially given that it does not require any link-state exchanges except for in the PPC phase.

For results discussed here, we have admitted a flow if bandwidth requirement of a flow satisfied (unless source-level flow control or trunk reservation is activated). Other requirements of a flow such as end-to-end delay requirement, packet loss rate requirement can also be included in our QoS routing computation framework quite easily. For each type of constraint, another filter can be added in the UPO phase to determine the pruned set of paths since the cached paths are already obtained from the PPC phase. While this increases computational time nominally in the UPO phase, newly arrived flows do not incur any additional flow set-up time delay in the ARC phase.

It is important to note that disparity in grade-of-service seen by heterogeneous service classes can not be completely addressed by QoS routing alone. For this, other measures such as source-based admission control and trunk reservation are needed. There are other challenges as well: in a sparsely-connected network, it is not easy to determine how to set the trunk reservation for a node-pair which is not directly connected. While there has been a lot of research on source-based flow admission control, a pragmatic scheme that can work in a general topology network running QoS routing (and trunk reservation) is still missing from the existing literature.

Overall, the notion of path caching is a viable option given that different routing schemes can be cast in the general framework and that other controls such as trunk reservation and source-based admission control can co-exist in the network as well. While we discuss our routing computation framework in a generic QoS routing environment, the basic idea can be applied to networking environments such as Multi-protocol label switching (MPLS) and GMPLS networks.

Appendix

Consider the source-destination router pair $[i, j]$. The set of cached paths for this pair determined at time τ (the PPC phase time window) is denoted by $\mathcal{P}_{[i,j]}(\tau)$. For path $p \in \mathcal{P}_{[i,j]}(\tau)$, let $\mathcal{L}_{[i,j]}^p(\tau)$ denote the set of links used by this path.

Let $A_\ell(t)$ be the available capacity of link ℓ at time t (obtained based on the link-state protocol for the URO phase). Then from a bandwidth availability perspective, the "cost" of the path p for $[i, j]$ is determined by the non-additive property of the available capacity on the bottleneck link along the path:

$$z_{[i,j]}^p(t) = \min_{\ell \in \mathcal{L}_{[i,j]}^p(\tau)} \{A_\ell(t)\}.$$

Since the path is known from the PPC phase, this filter operation is quite simple. If the index p is now renumbered in order of most available bandwidth to the least available bandwidth at time t , then we have:

$$z_{[i,j]}^1(t) \geq z_{[i,j]}^2(t) \geq \dots \geq z_{[i,j]}^{\#(\mathcal{P}_{[i,j]}(\tau))}(t).$$

Each source node can use the same principle to determine their own order path set based on periodic update of link-state information.

Similarly, the available bandwidth on a link as received through the link-state protocol is used by other source-destination pairs to determine their ordered paths periodically.

The availability of the bandwidth on a link is dependent on whether the trunk reservation is activated. Suppose the capacity of link ℓ is C_ℓ , and the currently occupied bandwidth as known at time t (based on link-state update) is $u_\ell(t)$. In the absence of trunk reservation, the available bandwidth on link ℓ is given by

$$A_\ell(t) = C_\ell - u_\ell(t).$$

If, however, a part of the link bandwidth $r_\ell(t)$ for link ℓ is for trunk reservation at time t , then

$$A_\ell(t) = C_\ell - u_\ell(t) - r_\ell(t).$$

The former is sometimes also referred to as residual bandwidth.

Acknowledgment

This work has been the result of many interactions with a number of students who worked with me over the years: Saravan Rajendran, Senthil Sankarappan, Sujit Guptan, Indrajanti Sukiman, Shishir Ramam, Raj Sivasankar, Shankar Subramaniam, Aekkachai Rattanadilokchai and Loren Rard. I'm thankful to Srinivasa Thirumalsetty for obtaining some of the recent computational results. The report benefited from many helpful comments from and discussion with Senthil Ayyasamy, Cory Beard, Padmanabhan Krishnan, Balaji Krithikaivasan, Marwan Krunz, Ibrahim Matta, Shekhar Srivastava, Appie van de Liefvoort. Jane Vogl's careful reading significantly improved the overall presentation of the paper.

Finally, this paper is dedicated to Jerry Ash from whom I first got the taste of network routing.

References

- [1] G. Apostolopoulos, R. Guerin, S. Kamat, S.K. Tripathi, "Quality of Service Based Routing: A Performance Perspective," *Proceedings of ACM SIGCOMM'98*, pp. 17-28, 1998.
- [2] G. Apostolopoulos, R. Guerin, S. Kamat, A. Orda, T. Przygienda, D. Williams, "QoS Routing Mechanisms and OSPF Extensions," Internet RFC 2676, August 1999. available from www.ietf.org
- [3] G. R. Ash, *Dynamic Routing in Telecommunication Networks*, McGraw-Hill, 1997.
- [4] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, "A Framework for QoS-based Routing in the Internet," Internet RFC 2386, August 1998. available from www.ietf.org
- [5] R. J. Gibbens, F. P. Kelly and P. B. Key, "Dynamic Alternate Routing — Modeling and Behavior," *Proc. 12th International Teletraffic Congress*, Torino, Italy, 1988.
- [6] B. Kraimeche and M. Schwartz, "Analysis of Traffic Access Control Strategies in Integrated Service Networks," *IEEE Trans. on Comm.*, Vol. COM-33, pp. 1085-1093, 1985.
- [7] R. S. Krupp, "Stabilization of Alternate Routing Networks," *Proceedings of IEEE International Conference on Communications (ICC82)*, pp. 31.2.1-31.2.5, Philadelphia, June 1982.
- [8] E. L. Lawler, *Combinatorial Optimization: Networks and Methods*, Holt, Rinehart and Winston, New York, 1976.
- [9] Q. Ma and P. Steenkiste, "On Path Selection for Traffic with Bandwidth Guarantees," *Proc. of Fifth IEEE Intl. Conf. on Network Protocols*, Atlanta, October 1997.
- [10] D. Medhi and S. Guptan, "Network Dimensioning and Performance of Multi-Service, Multi-Rate Loss Networks with Dynamic Routing," *IEEE/ACM Trans. on Networking*, Vol. 5, pp. 944-957, 1997.
- [11] D. Medhi and I. Sukiman, "Admission Control and Dynamic Routing Schemes for Wide-Area Broadband Networks: Their Interaction and Network Performance," *Proc. of Intl. IFIP-IEEE Conference on Broadband Communications*, Montreal, Canada, pp. 99-110, April 1996. For extended version, see: D. Medhi and I. Sukiman, "Multi-Service Dynamic QoS Routing Schemes with Call Admission Control: A Comparative Study," *Journal of Network & Systems Management*, Vol. 8, No. 2, pp. 157-190, June 2000.
- [12] R. J. Sivasankar, S. Ramam, S. Subramaniam, T. S. Rao and D. Medhi, "Some Studies on the Impact of Dynamic Traffic in QoS Based Dynamic Routing Environment," *Proc. IEEE Intl. Conf. on Communications (ICC2000)*, New Orleans, June 2000.

- [13] S. R. Thirumalsetty and D. Medhi, "On the Performance and Behavior of QoS Routing Schemes," Technical Report, University of Missouri-Kansas City, 2000.
- [14] J. Moy, *OSPF - Anatomy of An Internet Routing Protocol*, Addison-Wesley, 1998.
- [15] J. H. Weber, "A Simulation Study of Routing and Control in Communication Networks," *Bell Sys. Tech. J.*, Vol. 43, pp. 2639-2676, 1964.

Bio

Deep Medhi is Professor of Computer Networking and the Divisional Chair of the Computer Science & Electrical Engineering (CSEE) division in the School of Interdisciplinary Computing and Engineering (SICE) at the University of Missouri-Kansas City (UMKC). He received the B.Sc.(Hons.) degree in Mathematics from Cotton College, Gauhati University, India, the M.S. degree in Mathematics from the University of Delhi, India, and the M.S. and Ph.D. degrees in Computer Sciences from the University of Wisconsin-Madison in 1981, 1983, 1985 and 1987, respectively. Prior to joining UMKC in 1989, he was a member of the technical staff in the traffic network routing and design department at the AT&T Bell Laboratories, Holmdel, New Jersey from 1987 to 1989. He was an invited visiting professor in the Institute of Telecommunications at the Technical University of Denmark during the summer of 1999. He has published over forty five papers. His research interests are in survivable network design and architecture, dynamic quality-of-service routing, next generation network architecture, IP network availability, wireless networking, network management, and medical informatics. He is a member of the editorial board of the Journal of Network and Systems Management (JNSM). He has served on the technical program committees of several IEEE conferences including IEEE INFOCOM, and IEEE NOMS. He was a guest editor of the June 1997 special issue on "Fault Management in Communication Networks" of JNSM. In the past several years, his research has been funded by Defense Advanced Research Project Agency (DARPA), National Science Foundation (NSF), and Sprint Corporation.

He is the recipient of the Individual Performance Award at AT&T Bell Laboratories (1989), UMKC Trustees Award for Excellence in Teaching (1996), and UMKC Faculty Performance Shares Award (2001).

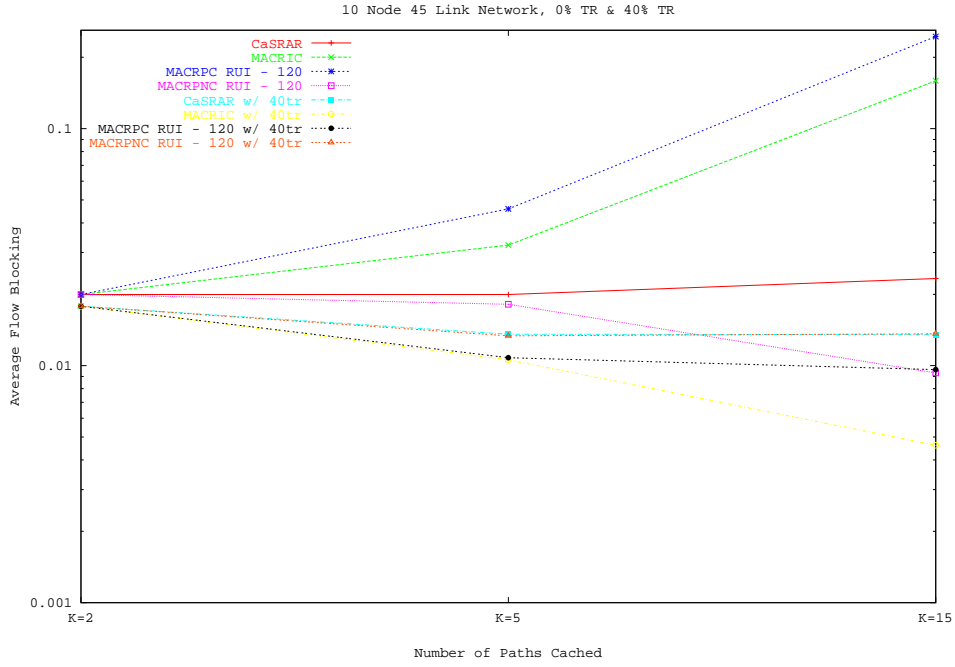


Figure 1: Homogeneous service fully-connected network (with and without trunk reservation)

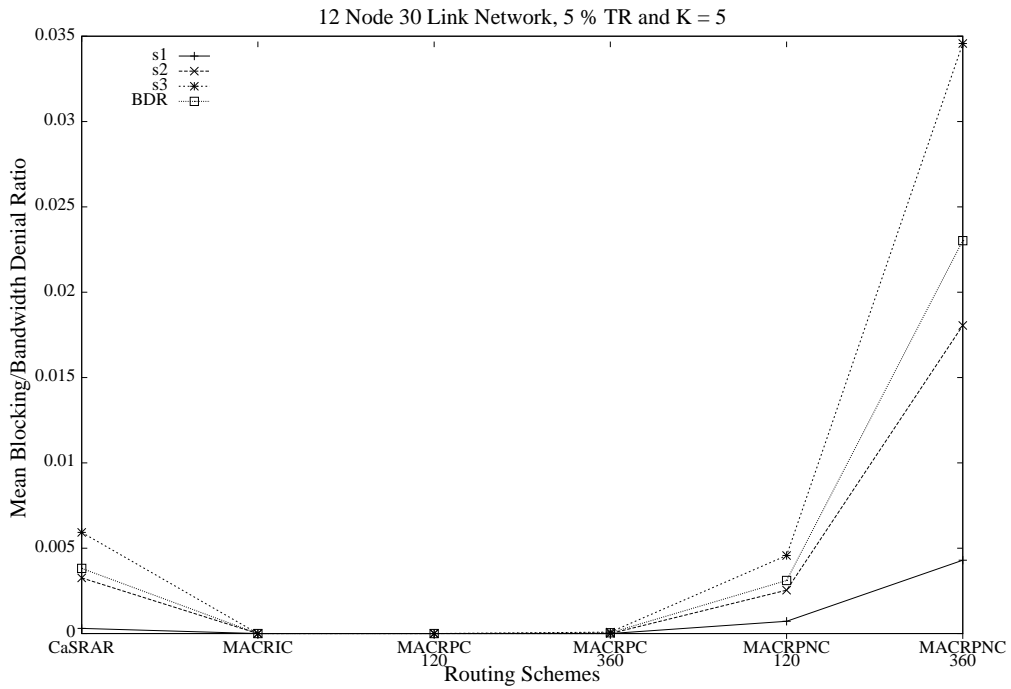


Figure 2: Performance of different routing schemes and update periods (in a sparsely-connected network, low-load case)

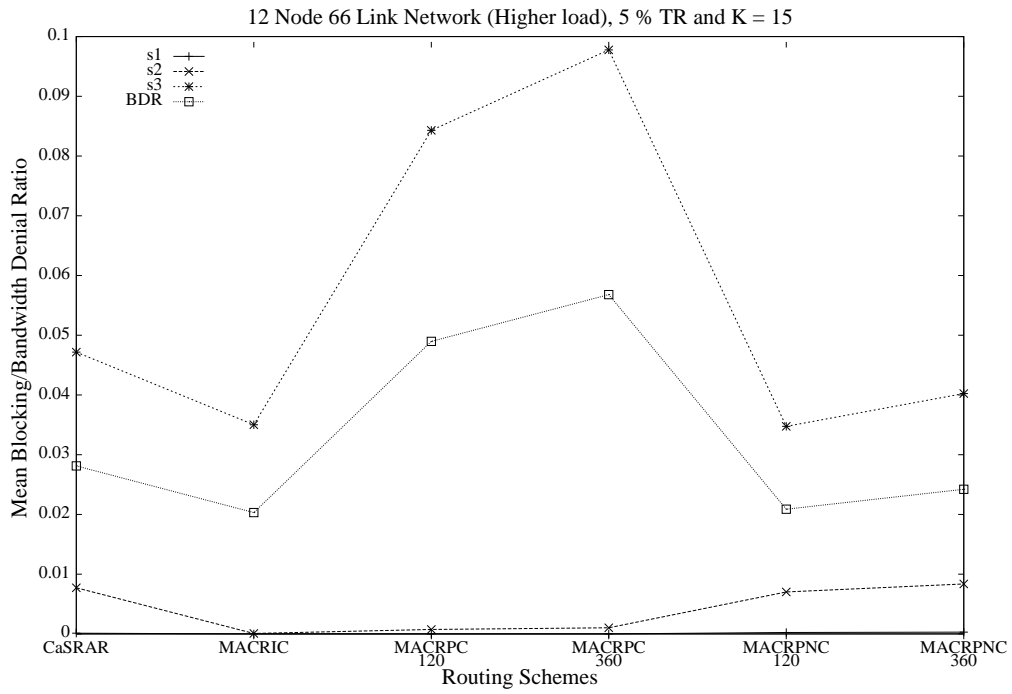


Figure 3: Performance of different routing schemes (and update periods), higher load case

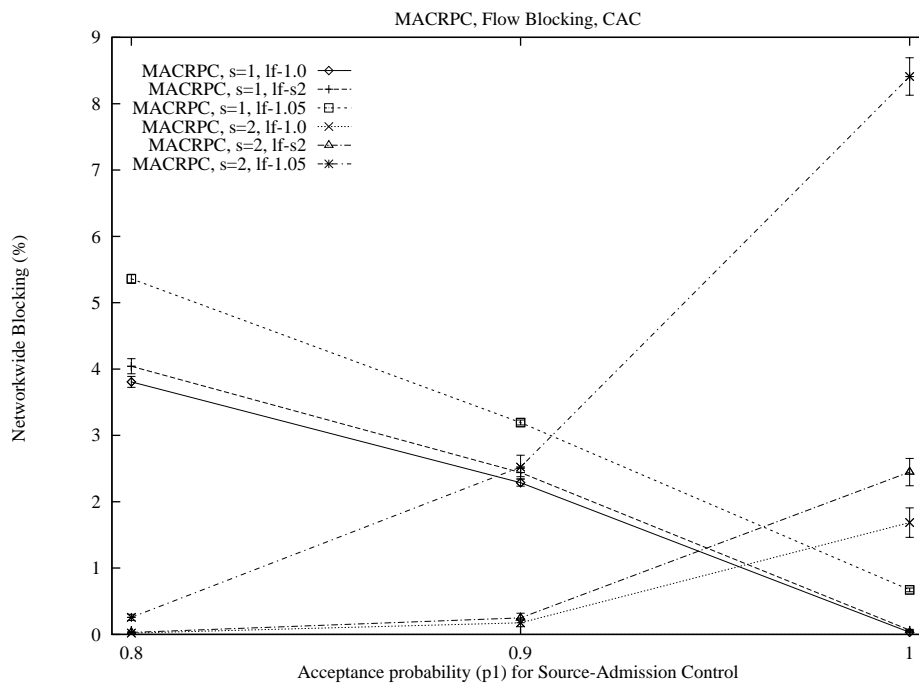


Figure 4: Performance Impact in the presence of source-based admission control

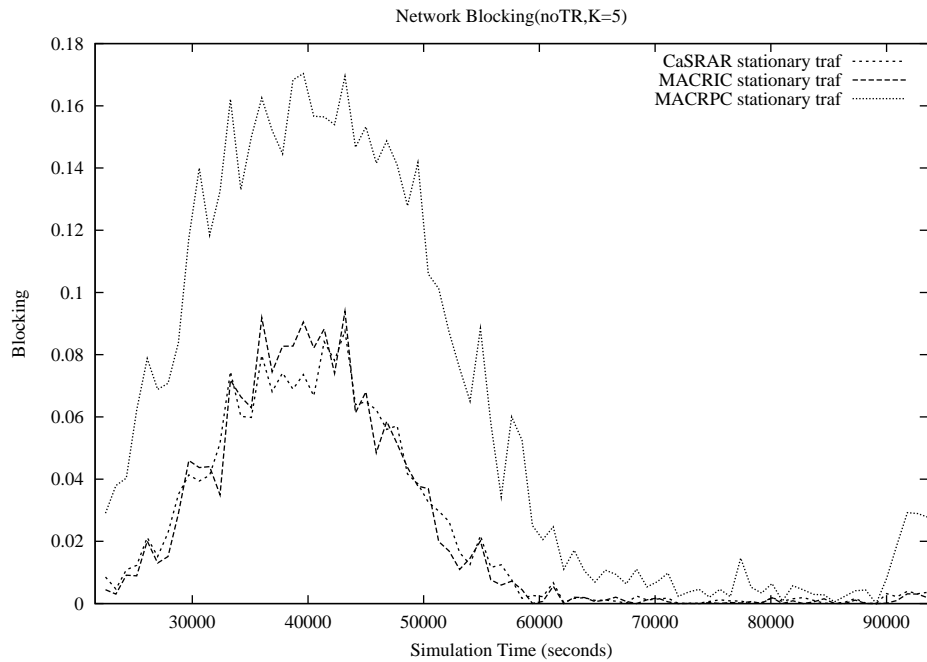


Figure 5: Dynamic Performance behavior of *stationary* traffic due to the influence of dynamic traffic (no trunk reservation)

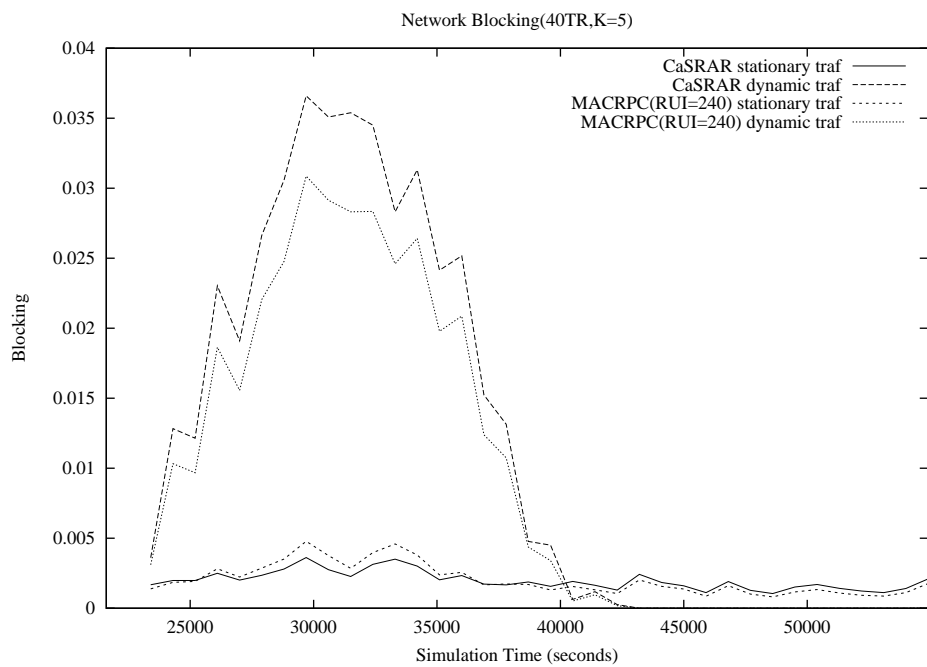


Figure 6: Performance of Dynamic and Stationary traffic (with trunk reservation)