A Study on using Network Flows in Hierarchical QoS Routing

Venkatesh Sarangan and Raj Acharya

Department of Computer Science and Engineering,
The Pennsylvania State University, University Park, PA 16802

Abstract- **QoS routing is the process of routing a connection based on the connection's resource requirements. The overhead involved in QoS routing increases with the network size.** *State aggregation* **is an important technique that helps to reduce the overhead. In this paper, we propose a new state aggregation technique based on "network-flow". Our approach allows a domain to update the aggregate sent by other domains and keep track of resource availability in other domains. We study the efficacy of our approach with respect to various network and traffic parameters. Preliminary simulation results show that our approach gives a better bandwidth admission ratio when compared with existing techniques.**

1. INTRODUCTION

Topology aggregation is an important technique for reducing the message overhead involved in QoS routing. Large networks like the Internet are composed of different autonomous systems or domains. A router that connects a domain to its neighbor is called as a border router (BR). In topology aggregation, each domain constructs an aggregate that consists of two parts: (a) aggregate of the connectivity and (b) aggregate of the resource availability in the domain. After constructing such an aggregate, a domain advertises it to others. The routers in the internetwork will have detailed information about their own domain and aggregated information about other domains. Inter-domain routing decisions are based on the aggregated information while intra-domain routing decisions are based on the detailed information. Hierarchical QoS routing is the process of selecting a path based on this mixture of detailed and aggregated state information.

The ATM PNNI standard [1] has proposed techniques such as *mesh*, *star* and *star with bypasses* for aggregating the connectivity. The border routers (BRs) in the domain form the vertices in the aggregation and are connected using logical links. While the PNNI standard proposes techniques for aggregating the connectivity, it does not prescribe any policy for aggregating the network state. For state aggregation, various policies could be followed. One such policy is to find the *best* path connecting the border routers in the domain and assign the metric of this best path to the corresponding logical link in the aggregated topology. For example, in aggregating bandwidth, the best path would correspond to the *widest* path and the metric of the best path would be the *bottle neck* bandwidth in the widest path. Hence in the aggregated topology, the logical link connecting two BRs will be assigned the bottle neck bandwidth in the widest path between the two BRs.

The rest of the paper is organized as follows. In section 2, we explain the existing schemes for topology aggregation. We outline the limitations in the existing schemes and the advantages of our approach in section 3. In section 4, we give the details of our aggregation scheme. We give our simulation results and discuss them in section 5. We finally conclude in section 6.

2. RELATED WORK

Various topology aggregation schemes have been proposed in [5], [6], [7], [8], and [9]. References [6] and [8] give some rules for assigning a QoS metric to the logical link. Reference [5] evaluates the two commonly used aggregation techniques - star and mesh and proposes new techniques for aggregating bandwidth, *hybrid aggregation* and *weighted aggregation*. References [7] and [9] give methods for simultaneously aggregating two metrics - *delay* and *bandwidth*. Aggregating two metrics is more complex because a path that is best for one metric need not be the best for the other metric too. To overcome this problem, reference [7] suggests that instead of having only the numerical values of bandwidth and delay, each logical link stores an additional parameter which implicitly defines a curve passing the bandwidth-delay pair on a delay-bandwidth plane. A drawback of the curve proposed in [7] is that the curve may be very far away from other parameters and could be independent of the quality of service we want. Reference [9] attempts to deal with this drawback by plotting the $<$ *bandwidth*, *delay* $>$ tuples of all unique paths between two BRs in the delay-bandwidth plane and approximate these points using a line segment. This line segment is then advertised to other domains.

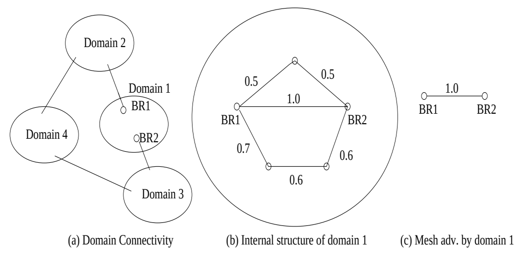

Figure 1: An example internetwork

3. LIMITATIONS IN EXISTING SCHEMES

Basically all these approaches assign the metric of the best path or the metric of a combination of paths between the BRs to the logical link in the topology aggregate. Inherent to a domain is its finite capacity to route traffic. The existing approaches do not consider the capacity of a domain while routing a request through it. If we do routing, taking into consideration, the traffic capacity of a domain, the performance can improve. This can be easily understood by considering the example network shown in figure 1.

It shows a network with four domains *1*, *2*, *3* and *4* . The internal structure of domain *1* is shown. The numbers indicate the bandwidth available in the links. The topology aggregate of domain *1* is shown in figure 1c. Domain *1* advertises this aggregated information to its neighboring domain *2*. It is clear that at any instant, domain *1* cannot accept more than *2.2* units of traffic from domain *2*. Let domain *2* get five connection requests to domain *3*, each for *0.5* units of bandwidth. Based on the aggregate sent by domain *1*, domain *2* decides to forward these requests to domain *1*. If these five requests arrive in quick succession before domain *1* can send its new update, the last request would fail. This is because domain *2* would still think that domain *1* can support the fifth request while it cannot. Thus routing without taking into consideration the traffic capacity of a network brings down the performance. In this paper, we propose to use the traffic a network can support as an aggregated metric. A domain would transmit the traffic along with the star or mesh aggregate.

The advantage of sending the traffic as a metric is that, it helps domain *2* to keep track of the resources available in domain *1* with reasonable accuracy without having to receive updates from it. If domain *2* can decide that domain *1* cannot support a call, it can look for alternate paths to route the call. Providing a metric like traffic is clearly necessary, since in hierarchical QoS routing, inter-domain updates are exchanged less frequently. Taking routing decisions using stale information for a long time can bring down the routing performance considerably. A domain can calculate the traffic it can support by calculating the *maximum flow* though it. Section 4 explains how the maximum flow problem could be applied for calculating the traffic capacity of a domain.

The most relevant work in using network flows for QoS routing is by Rao and Batsell [11]. They used network flows for solving the *Message Transmission problem (MTP)* and to develop a heuristic for the NP-complete *Sequence transmission problem (STP)*. Their approach allows a connection to be established over multiple paths. We propose to use network flows for inter-domain routing and we do not allow a connection to be routed over multiple paths. To the best of our knowledge, no techniques have been proposed so far to use network-flows for inter-domain QoS routing.

4. TOPOLOGY AGGREGATION USING NETWORK FLOWS

In this paper we assume that routers within a domain maintain a detailed intra-domain link state information by exchanging LSAs. Also a domain advertises its topology aggregate only to its neighbors and this advertisement is not forwarded beyond the immediate neighborhood.

4.1. Traffic calculation

A detailed analysis of the max-flow problem could be found in [2], [3]. Calculating the traffic a domain can support from its neighbor is nothing but the *max-flow* problem with multiple sources and multiple sinks. The sources and sinks for the flow and the capacities for the edges in the flow graph should be determined. The following paragraph explains how this is done.

Figure *2* shows a portion of an internetwork. Domain *j* is connected to four neighboring domains *I*, *II*, *III* and *IV* through the border routers *BR1*, *BR2*, *BR3* and *BR4* respectively. Domain *j* should advertise to its neighbors the amount of traffic it can accept from them. Let us calculate the traffic value that has to be advertised to domain *I* at time instant *t*. The traffic from domain *I* enters domain *j* through border router *BR1*. The incoming traffic can leave domain *j* through any of the other border routers *BR2*, *BR3* or *BR4*. Hence to calculate the traffic that could be accepted from domain *I*, *BR1* acts as the source and the border routers {*BR2*, *BR3*, *BR4*} act as the sinks; Domain *j*'s topology becomes the flow network. The capacity of an edge in the flow network is nothing but the available bandwidth in the corresponding physical link at instant *t*. This value is readily available since, routers within a domain maintain a detailed link state information by exchanging LSAs. Thus no additional message exchanges are required to calculate the edge capacities in the flow network.

Traffic calculation deviates slightly from the multi-source, multi-sink problem as follows: In the multi-source, multi-

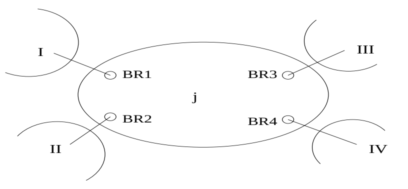

Figure 2: A portion of an internetwork

sink problem, the individual sinks are connected to the super-sink using edges of infinite capacities. In calculating the traffic, the individual sinks are connected to the super sink by using edges of appropriate capacities. In our example, the edge between the sink *BR2* and the super sink will be assigned a capacity equal to the traffic aggregate advertised by domain *II* to domain *j*. Similarly, the link connecting the sink *BR3* and the super sink will be assigned a capacity equal to the traffic aggregate advertised by domain *III* to domain *j* and so on. Thus all the parameters in the flow network are known. Border router *BR1* calculates the traffic using the *Ford-Fulkerson* algorithm with the above said parameters of the flow network and advertises to domain *I*.

In the same way, to calculate the traffic value that can be accepted from domain *II*, *BR2* acts as the source and the border routers $\{$*BR1, BR3, BR4*$\}$ act as the sinks. The capacities of all the edges are assigned as explained above and the traffic is calculated.

4.2. Centralized vs Distributed Max-flow Computation

The border routers use a centralized algorithm to do the flow computation. Distributed flow computation algorithms based on [4] could also be used, but the signaling overhead involved in such approaches is significant. Also, they tend to be more complex to implement than the centralized algorithms. A distributed approach is used for flow calculation if (a) the information about the entire flow network is not available and/or (b) the network topology and the link capacities change very frequently. In our approach, the routers maintain a detailed state information about all the links in the network. We assume that this link state information stored by the routers is never stale. The network topology too, remains fairly unchanged. Hence a distributed approach is not warranted.

4.3. Routing

In this section, we describe the routing algorithm used in our work. All routers in the network are assumed to know the domain connectivity information i.e. the neighborhood information of all the domains. Since, this is only based on the physical connectivity, it is easy to obtain this information and the overhead involved in obtaining this information is less. A domain stores the aggregate information of its neighbors alone. We restrict the scope of this inter-domain state information to neighbors alone because, it is not meaningful for a domain to update the traffic aggregate sent by another domain that is several domain hops away. Since a domain has information about only its neighbors, we do *distributed inter-domain* routing. Within a domain we do *source routing*. A feasible path is defined as the one that can support the requested bandwidth.

The trivial way of doing distributed inter-domain routing is to do flooding between domains. However this results in an enormous amount of un-necessary overhead. An elegant way would be to follow the shortest path in terms of *domain-hops* to the destination domain, if that path can support the request. If the shortest path is not capable of supporting the request, the search branches off at that point and other possible paths are searched. This generates very less overhead when compared to blind flooding.

For simplicity, in the following discussions we shall focus on a single request. Each connection request is identified by an unique connection identifier *cid*. On receiving a connection request, the source router identifies the destination domain. The router then finds out the *shortest-path (SP)* in terms of domain-hops to the destination domain. The request is then source-routed along a feasible path to the egress router connected to the next-hop domain in the *SP*. The egress router checks whether the next-hop transit domain can support this request. A domain is said to support the request if two conditions are satisfied: (a) the bandwidth requested should be less than the bandwidth of the widest path and (b) the bandwidth requested should also be less than the current traffic estimate of the domain. If the next-hop domain can support the request, the request is forwarded to the ingress router in the next-hop domain. The ingress router in the transit domain, then identifies its next-hop domain and checks if that domain can support the request. If so, the request is source-routed to the corresponding egress router along a feasible path. If a domain say d_{k-1} along the shortest domain-hop path $\{d_1, d_s, \cdots, d_{k-1}, d_k, \cdots, d_l\}$ finds that the next domain d_k is not able to support the request, then domain d_{k-1} floods the request to its other neighbors. Each neighboring domain on receiving the request, tries to route the request along the shortest domain-hop path to the destination. This process goes on until the ingress router in the destination domain is reached. The ingress router in the destination domain routes the connection to the destination along a feasible path. The destination accepts the first copy of the request and sends an *ack* back. The destination rejects all duplicate connection requests. The ack traverses through the same domains as the connection request. As the ack is propagated upstream, reservations are made. The connection is established when the ack reaches the source. Since we use a distributed scheme it is quite possible that loops

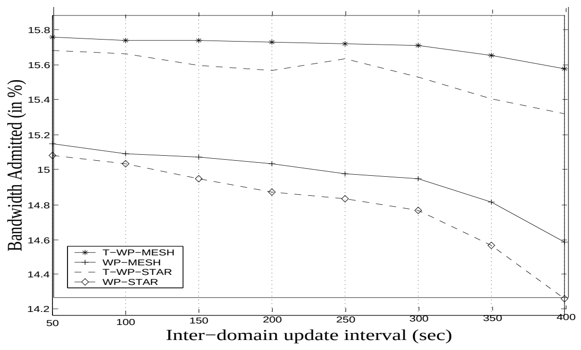

Figure 3: Bandwidth admitted vs varying inter-domain update intervals. The capacity of intra-domain and inter-domain links were respectively 17.5 Mbps and 60 Mbps.

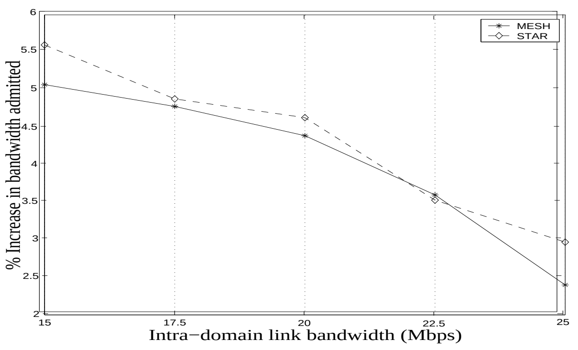

Figure 4: % Increase in bandwidth admitted by *T-WP* over *WP* for various link capacities

could be created. Each connection request carries with it, the list of domains it has visited so far. Hence, by reading the list, loops could be detected. Requests that go around in loops are discarded.

5. EXPERIMENTAL RESULTS

Simulations were done on networks of various sizes to test the efficacy of using the traffic as an aggregated QoS metric. The results reported in this paper are from a 30 domain network. The total number of routers in the network is 650. The simulations were done using OPNET, a commercial simulation software. All the links in the network are assumed to be duplex. The results shown are the average of several simulation runs. The connection requests arrive as per a Poisson process. The call durations are from an exponential distribution and the bandwidths requested are from an uniform distribution. During the simulations, the intra-domain link state exchange interval is kept constant. Each simulation is run for *25000* inter-domain connection requests. Using our distributed routing, the performance of the two aggregation schemes: (a) Advertising the widest path along with the traffic *(T-WP)* and (b) Advertising the widest path alone *(WP)* are compared. The performance of the two approaches is compared in terms of *bandwidth-admission ratio*. Bandwidth admission ratio is defined as the ratio of total bandwidth admitted to the total bandwidth submitted [10].

5.1. Sensitivity to Routing update interval

The bandwidth admitted into the network by *T-WP* and *WP* is shown in figure 3. It also shows the performance of advertising the widest-paths using *mesh* and *symmetric star* configurations. The bandwidth available for reservation in the intra-domain links is set as *17.5Mbps*. From the graph we can readily notice that using traffic as an aggregate metric increases the bandwidth admitted in to the network. Also, we can see that frequent routing updates improve the routing performance.

An important observation is that as the update frequency is reduced, the performance of *WP* goes down much faster than that of *T-WP*. The percentage improvement achieved by *T-WP* over *WP* varies from *4-7%* as the update interval is varied from *50* to *400* seconds. So, as the inter-domain update interval is increased, we gain more by using *T-WP*. This is because, even under long update intervals, a domain can estimate the resource availability in its neighbors using the traffic. Estimating the resource availability in other domains helps in taking better routing decisions and makes the routing process less sensitive to routing update intervals. In real life, it is desirable to exchange information outside a domain less frequently. Hence, it is better to advertise traffic also under such circumstances.

5.2. Sensitivity to Resource availability

The intra-domain link bandwidth was varied and the performance of *T-WP* and *WP* under different link capacities were studied. For clarity, from now on we shall show only the percentage increase in bandwidth admitted by *T-WP* over *WP*. Figure 4 shows the improvement achieved by *T-WP* (mesh and star) over the corresponding *WP* (mesh and star) at different link capacities. In these simulations, the inter-domain link capacities were kept at a much higher value than the intra-domain link capacities. The ratio of inter-domain link capacity to intra-domain link capacity was kept at *3*. So, as the intra-domain link capacities are varied, the inter-domain link capacities were also suitably varied to maintain the ratio. From the graph, we can see that, as more and more resources are available, the gains achieved by using *T-WP* goes down. This is not surprising since, when more resources are available, a 'blind' path selection is as good as a 'good' path selection.

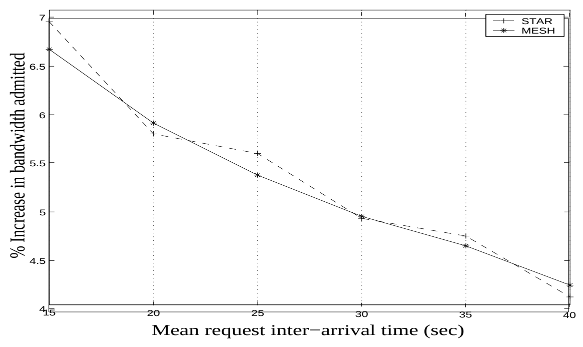

Figure 5: *%* Increase in bandwidth admitted by *T-WP* over *WP* for various request arrival rates

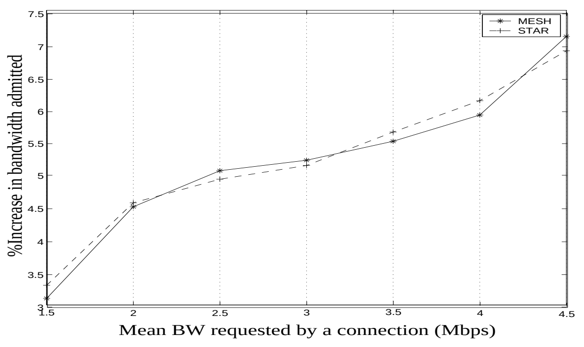

Figure 6: *%* Increase in bandwidth admitted by *T-WP* over *WP* for various requested bandwidths

5.3. Sensitivity to Traffic pattern

Figure 5 shows the percentage increase in bandwidth admitted by *T-WP* (mesh and star) over *WP* (mesh and star) for various request arrival rates. The gains obtained by *T-WP* increase as the request arrival rate increases. This is because, at high request arrival rates, the network bandwidth gets used up very rapidly. So, the aggregate sent in *WP* quickly becomes out of date. Hence the performance of *WP* goes down at high arrival rates. However, the traffic aggregate sent in *T-WP* helps a domain to keep track of the resource availability in neighboring domains. So, it does not suffer as much as *WP* at high request arrival rates. Thus we gain more by using *T-WP* over *WP* as requests come faster and faster.

Figure 6 is similar to figure 5 and shows the gains in using *T-WP* over *WP* as the mean bandwidth requested by a connection is varied. We see that as a connection requests more and more bandwidth, we gain more by using *T-WP*. The reason is same as before. As a connection asks for more and more bandwidth, with each connection establishment/tearing down, there is a greater fluctuation in the resource availability. *T-WP* keeps track of these fluctuations using the traffic estimate and hence performs better when the fluctuations are more.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a new topology aggregation scheme for aggregating bandwidth based on "network flows". Our scheme enables a domain to keep track of resource changes in other domains without receiving any updates. No additional message overhead is incurred in our approach. Simulation results show that our scheme has *4%-7%* performance gain over existing approaches when the inter-domain update interval is varied from *50* to *400* seconds. Also the performance of our scheme improves as the load on the network becomes heavier. In future, we would like to extend this network flow approach to cases where bandwidth is modeled as a random variable. We believe that such an approach will further increase the routing performance. We also would like to explore the feasibility of extending this network-flow approach for aggregating *additive metrics* like delay, jitter and cost.

REFERENCES

[1] Private network-network interface specification. version 1.0. *ATM Forum Technical report af-pnni-0055.000*, March 1996.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, 1990.

[3] L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, New Jersey, 1962.

[4] A. Goldberg and R. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM*, pages 921–940, 1988.

[5] F. Hao and E. Zegura. On scalable qos routing: Performance evaluation of topology aggregation. In *INFOCOMM 2000 Proceedings*, 2000.

[6] A. Iwata, H. Suzuki, R. Izmailow, and B. Sengupta. Qos aggregation algorithms in hierarchical atm networks. In *IEEE Proceedings of the ICC'98*, 1998.

[7] T. Korkmaz and M. Krunz. Source-oriented topology aggregation with multiple qos parameters in hierarchical atm networks. *to appear in ACM Transactions on Modeling and Computer Simulation*.

[8] W. Lee. Topology aggregation for hierarchical routing in atm networks. In *ACM SIGCOMM'95 Proceedings*, 1995.

[9] K.-S. Lui and K. Nahrstedt. Topology aggregation and routing in bandwidth-delay sensitive networks. In *IEEE Globecom 2000 Proceedings*, 2000.

[10] Q. Ma and P. Steenkiste. On path selection for traffic with bandwidth guarantees. In *Fifth IEEE International Conference on Network Protocols, Atlanta*, 1996.

[11] N. Rao and S. G. Batsell. Qos routing via multiple paths for bandwidth reservation. In *IEEE INFOCOMM'98 Proceedings*, 1998.