Practical Multipath Load Balancing with QoS

Bradley R. Smith
Computer Engineering Department
Jack Baskin School of Engineering
University of California Santa Cruz
Email: brad@soe.ucsc.edu

Lincoln Thurlow
Computer Science Department
Jack Baskin School of Engineering
University of California Santa Cruz
Email: lthurlow@soe.ucsc.edu

***Abstract*—The Internet is based on a single-path communications model. This model imposes significant constraints on the ability of the Internet to satisfy the quality-of-service requirements of network applications, and results in significant inefficiencies in the use of network resources that are manifested as congestion. The result has been the need to over-provision Internet-based systems to meet the basic needs of modern communications. With the adoption of the Internet as the converged communication infrastructure for the 21st century, this is clearly not an acceptable long-term solution.**

One approach that has been identified to address these limitations is to enhance the Internet routing architecture to support multiple paths between a given source and destination. Significant research has been done into multi-path solutions for QoS and congestion, however a comprehensive solution for both QoS and congestion that is compatible with the Internet's datagram, hop-by-hop model of communication is still elusive.

This paper reviews a solution presented in previous work, called Dominant Set Multipath Routing (DSMR), that addresses these requirements. The DSMR algorithm computes the *best set of routes* between each source and destination that provides the full range of performance available from the network. This set is used to route flows over paths that both meet the QoS requirements of the flow and minimize congestion in the network. Simulations are then presented which show the effectiveness of DSMR to provide 3 to 11 times the capacity of single-path routing while meeting QoS requirements and minimizing congestion.

I. Introduction

The Internet is based on a *best-effort* communication model in which the "best" path is pre-computed by each router to all destinations (triggered by topology changes), and packets are forwarded on a best-effort basis (i.e. they may be dropped or delivered out-of-order). Packet forwarding is implemented on a *hop-by-hop* basis where forwarding tables are computed independently at each router, and the forwarding decision is done on a per-packet basis.

The current Internet uses a *destination-based* form of hop-by-hop forwarding that only supports a single path to each destination. Specifically, Internet forwarding state is composed of a single entry for each destination in the internet, giving the next-hop router on the best path to the destination. As a result, only one path is supported to any given destination, and that path is computed to optimize a single metric.

The use of single-path routing significantly compromises the ability of a network to meet the *quality-of-service* (QoS) requirements of diverse applications, and tends to result in poor utilization of network resources.

Support for the QoS requirements of a diverse set of network applications requires, in general support for multiple paths between a given source and destination. A number of metrics can be used to quantify the performance of a communications network. For example *latency* is a measure of the delay traffic experiences as it traverses a network, *jitter* is a measure of the variation in that delay, *bandwidth* is a measure of the rate at which data can pass through a point in a network, etc.

Many applications have special requirements of the network they run on [1]. For example interactive audio (i.e. VoIP) requires low latency and jitter of its communication channel to support natural, conversational interaction, however it has relatively minimal bandwidth requirements. In contrast, video streaming requires high bandwidth and low jitter to provide a smooth viewing experience, however it has relatively minimal latency requirements (it's OK if the video takes a number of seconds to start, as long as it runs smoothly once it starts).

Satisfying constraints on multiple metrics requires, in general, the use of multiple paths between any two nodes. For example, given two paths between two nodes with the following parameters: path 1 has bandwidth of 100Kbps, latency of 20ms, and low jitter; path 2 has bandwidth of 2Mbps, latency of 200ms, and low jitter; which is the better path? Path 1 would be preferred for an interactive audio application while path 2 would be preferred for video streaming. With multiple metrics, the preferred path depends on the requirements of the application.

Single-path routing has a similarly detrimental effect on the utilization of network resources. As the load in a network increases, sending all traffic between a given source and destination over a single path tends to result in links on that path becoming congested. The hop-by-hop style of packet forwarding used in the Internet exacerbates this problem.

With destination-based forwarding each router forwards packets by matching each packet's destination address with a single entry in the router's forwarding table. This leads to the constraint that all traffic forwarded through an intermediate router to a destination must follow the same path used by traffic sent from that router to the destination. This aggressive tendency to concentrate traffic on a subset of a network's topology causes traffic to experience congestion while usable network resources are left idle, resulting in poor utilization of network resources.

The result is the Internet we have today, which is challenged to meet the QoS requirements of an increasingly diverse set of network applications, requiring significant over-provisioning to provide acceptable levels of service. As the foundation for the converged communication infrastructure of the twenty-first century [2], this is clearly not acceptable.

Single-path routing involves two processes: *route computation* and *packet forwarding*. Multi-path routing requires modifications to these processes to support multiple paths per destination, and the addition of a *path selection* process to choose a path to use for specific traffic.

Destination-based forwarding can't support multi-path routing, and therefore must be replaced by a more general forwarding mechanism. As will be discussed later, we propose the use of tag-switching as the only solution that supports multi-path routing while retaining the Internet's distributed, hop-by-hop forwarding architecture.

In previous work [3] we have presented a new multi-path routing architecture based on the concept of a *best set of routes* to each destination, and route computation algorithms that efficiently compute this set. This architecture is compatible with the Internet's best-effort communication model, and provides a framework for delivering the full range of performance available in a network.

In the remainder of this paper we review this architecture and present the results of simulations that quantify the capacity gains it provides. The primary goal of this work is to determine, to a first approximation, the benefits available from this routing architecture. Section II presents previous work in the use of multiple paths to improve support for QoS and minimize congestion. Section III presents the DSMR algorithm. Section IV presents the simulations. Lastly, Section V presents our conclusions.

II. Previous Work

This section presents previous work on improving support for QoS and the use of multiple paths for both QoS and congestion control. Each solution is assessed in terms of whether it addresses both QoS and congestion control, and how compatible it is with the Internet architecture in terms of implementing a best-effort communication model (pre-computation of routes and hop-by-hop forwarding).

Two enhancements to the Internet architecture to support QoS have been proposed representing fundamentally different approaches to solving the problem of resource management in the context of performance requirements, the Intserv and Diffserv architectures.

The goal of the *integrated services* (Intserv) architecture [2] is to define an integrated Internet service model that supports best-effort, real-time, and controlled link sharing requirements. Intserv makes the assumption that network resources must be explicitly controlled, and defines an architecture where applications reserve the network resources required to implement their functionality, and an infrastructure of admission control, traffic classification, and traffic scheduling mechanisms which implement the reservations. In the Intserv architecture resource reservations are sent along paths computed by the existing routing infrastructure. As a result, requests may be denied when resources do not exist along the current route when in fact paths exist that could satisfy the request. Intserv is based on a virtual-circuit communications model and, therefore, has all the limitations of that model relating to robustness, efficiency, and responsiveness.

In contrast, the *differentiated services* (Diffserv) architecture [4] provides resource management without the use of explicit reservations. In Diffserv, a small set of *per-hop forwarding behaviors* (PHBs) is defined within a Diffserv domain which provide resource management services appropriate to a class of application resource requirements. Traffic classifiers are deployed at the edge of a Diffserv domain that classify traffic for one of these PHBs. Inside a Diffserv domain, routing is performed using traditional hop-by-hop, address-based forwarding mechanisms.

Diffserv retains the best-effort, distributed, hop-by-hop, datagram routing model of the Internet, and therefore retains the robustness, efficiency, and responsiveness of the Internet. However, similar to the Intserv model, communications resources to a given flow in a Diffserv environment are limited to those available along the paths computed by the existing routing infrastructure. As a result QoS requirements may not be satisfied when adequate resources are not available along the current route when in fact paths exist that could satisfy the requirements.

In addition, there has been extensive research into solutions for reducing congestion through the use of multiple paths to each destination.

Vutukury and Garcia-Luna [5] present an approximation to Gallager's minimum-delay routing algorithm [6]. The solution pre-computes multiple paths of unequal length to each destination, along with an allocation of traffic to each path. The primary goal of the algorithm is to minimize the delay traffic experiences as it traverses the network. In this work traffic is forwarded along different paths without regard to the flow it is a part of. To address the problem this causes for TCP traffic a later paper [7] presents a solution that allocates individual TCP flows to a single path. This solution precomputes paths and uses hop-by-hop forwarding, however its focus is minimizing delay.

Taft-Plotkin et al [8] present a solution for using multiple paths to meet the QoS requirements of flows. A fixed number of paths are precomputed that include maximally disjoint paths with minimum delay and maximum bandwidth. These paths are sorted by available bandwidth, and paths are selected by an ordered search of the list for the first path which satisfies the QoS requirements of the flow. The solution depends on per-flow path setup with admission control. The goal of the algorithm is to satisfy the QoS requirements of flows while minimizing congestion.

Nelakuditi and Zhang [9] present a solution for minimizing congestion in a network by forwarding traffic over multiple paths. The solution computes a set of widest-shortest paths to each destination, where the size of the set is a parameter of the computation. Traffic is then allocated to these paths based on the offered load and blocking probability observed locally for each path. This solution pre-computes paths, but depends on path-setup for forwarding traffic, and does not attempt to satisfy QoS requirements of flows. One interesting result from the simulations presented in the paper is that only a small number of paths are needed for near optimal call blocking performance.

There has been extensive research into the use of multiple paths for a TCP flow [10], [11]. While the goal of these efforts is similar to that of DSMR, to exploit bandwidth available along multiple paths in a network between a source and destination, the approach is both different and complimentary to that of DSMR. The challenge addressed by multi-path TCP is to enhance TCP to allow traffic for a single flow to be sent over multiple paths to a destination, while avoiding the penalties incurred by existing TCP congestion avoidance mechanisms in response to the increased likelihood of segments being delivered out of order. As implemented in the simulations presented here, DSMR assigns one flow to a single path, and exploits multiple paths by assigning multiple flow between a given source and destination to different paths. DSMR is complimentary to the multi-path TCP solutions in that a DSMR environment would provide multiple paths to the multi-path TCP mechanisms that simultaneously meet the QoS needs of the application, and minimize congestion. This integration is an area for future research.

There has also been extensive research into mechanisms for selecting one path among a candidate set of paths with the goal of minimizing congestion in the network [12], [13], [14]. These solutions use different techniques for detecting increasing congestion along different paths in a network. This information is used at path selection time to favor less-congested paths for new traffic. These solutions are complimentary to DSMR in that DSMR computes a comprehensive set of routes (in terms of QoS) for use by these path selection mechanisms.

A characteristic shared by these solutions is they all implement a *distributed* solution to path selection. An alternative approach, used in the simulations presented here, is centralized path selection where information is maintained describing the bandwidth available on each link in the network and flows are assigned to paths with adequate capacity whenever possible. The effectiveness of this centralized approach is limited by the fidelity of its link state information. This centralized approach is used in the simulations to provide an accurate, best-case scenario for the DSMR architectural model. This centralized architecture can be implemented in practice using the software defined networking model implemented in OpenFlow [15], where a centralized network controller maintains a global view of the current network state. This is another area for further research.

In summary, there has been extensive research into the use of multiple paths to minimize congestion, satisfy QoS requirements of flows, and occasionally to do both. However there appears to be no work that addresses the need for a comprehensive, multipath solution to congestion and QoS that is consistent with the Internet architecture's use of pre-computed routes and hop-by-hop forwarding. The remainder of this paper evaluates a general approach to such a solution.

To characterize the potential of this approach we present the results of high-level simulations where routes are computed using the DSMR algorithm, presented below, based on static link metrics (specifically total bandwidth and fixed link delay). These simulations assume a centralized route computation with knowledge of the topology and loads on the links.

III. Dominant Set Multipath Routing

As discussed in the Introduction, satisfying constraints on multiple metrics requires, in general, the use of multiple paths between any two nodes in a network. This correspondence between multiple metrics and multiple paths can be described formally by representing the set of metrics used to describe the performance of paths from a given source and destination pair as points in a multidimensional space. We'll call such a set of multiple metrics a link or path *weight*. Figure 1 plots the weights of 9 paths between a specific source and destination in an example network where the metrics composing the weights are bottleneck bandwidth and latency. "Better" values of these metrics are towards the origin of the graph (i.e. a perfect path would have infinite bandwidth and 0 latency).

These points can be interpreted as representing a region, up and to the right (away from the origin) of QoS values that each weight *satisfies* in the sense that the path represented by the weight would satisfy any QoS requirement in that region of the graph. Figure 2 depicts the regions satisfied by each path. Note that regions satisfied by some of the paths are fully contained in the regions of other paths. In the figure these *dominated* regions are represented with dashed lines.

A *best set* of paths to the destination can be identified as the set of paths that are not dominated by another path. This set of paths is *best* in the sense that any QoS requirement that is satisfiable by an existing path between the given source and destination, is satisfiable by a path in this set. We call these regions the *performance classes* available from the network for the destination. Figure 3 shows the performance classes for the example network.

The goal of QoS routing is to compute paths in a network that satisfy the performance requirements, expressed in terms of constraints on multiple metrics, of applications communicating across the network. The formalism presented above shows that, by definition, QoS routing must support the use of multiple paths between a source and destination.

Based on this notion of a dominant set of routes we can articulate a comprehensive multipath solution as one where flows are forwarded over paths in the dominant set that both meet their QoS requirements and avoid creating congestion in the network, if such paths exist.

This is the first routing solution that can make this commitment to avoid congestion and satisfy QoS requirements, if at all possible in the network. The primary insight motivating this new routing architecture is viewing weights of the set of paths to a destination as a partially ordered set, and computing

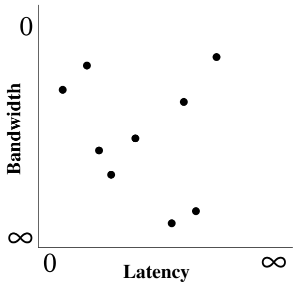


Fig. 1. Path Weights

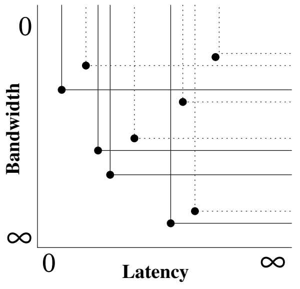


Fig. 2. QoS Regions

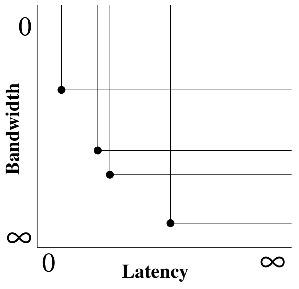


Fig. 3. Performance Classes

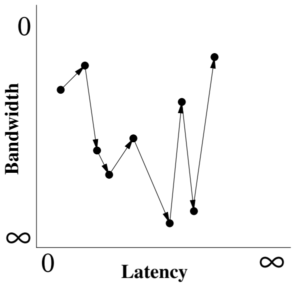


Fig. 4. Total Ordering

the dominant set of weights for this partial order as the foundation of a forwarding table. We call this model *Dominant Set Multipath Routing.*

One challenge of this solution is the need to forward traffic over multiple paths between a given source and destination in a hop-by-hop manner. Alternatives include source routing and *tag-switching* [16]. Source routing is an inherently centralized packet forwarding mechanism, therefore we propose the use of tag-switching to provide this capability. However, due to space constraints we don't describe this solution here. See [3] for details.

A. The DSMR Algorithm

As described above, *path weights* are composed of multi-component metrics that capture all important performance measures of a link such as delay, delay variance ("jitter"), available bandwidth, etc. The best set of paths to a destination is defined using an enhanced version of the path algebra defined by Sobrinho [17].

Formally, the path algebra $P = < W, \oplus, \preceq, \sqsubseteq, \overline{0}, \overline{\infty} >$ is defined as a set of weights W, with a binary operator $\oplus$, and two order relations, $\preceq$ and $\sqsubseteq$, defined on W. There are two distinguished weights in W, $\overline{0}$ and $\overline{\infty}$, representing the least and absorptive elements of W, respectively. Operator $\oplus$ is the original path composition operator, and relation $\preceq$ is the original total ordering from [17], which is used to order the paths for traversal by the path selection algorithm. Operator $\oplus$ is used to compute path weights from link weights. The routing algorithm uses relation $\preceq$ to build the forwarding set, starting with the minimal element, and by the forwarding process to select the minimal element of the forwarding set whose parameters satisfy a given QoS request.

A new relation on routes, $\sqsubseteq$, is added to the algebra and used to define classes of comparable routes and select maximal elements of these classes for inclusion in the set of forwarding entries for a given destination. Relation $\sqsubseteq$ is a partial ordering (reflexive, anti-symmetric, and transitive) with the following, additional property:

Property 1: $(\omega_x \sqsubseteq \omega_y) \Rightarrow (\omega_x \succeq \omega_y)$.

A route r_m is a *maximal element* of a set R of routes in a graph if the only element $r \in R$ where $r_m \sqsubseteq r$ is r_m itself. A set R_m of routes is a *maximal subset* of R if, for all $r \in R$ either $r \notin R_m$, or $r \in R_m$ and for all $s \in R - \{r\}$,

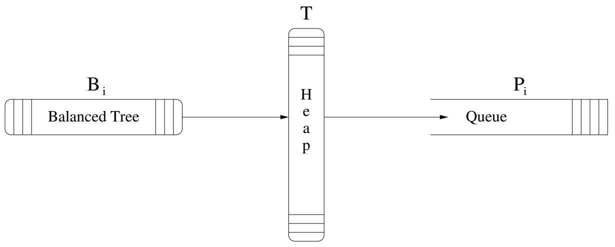


Fig. 5. Data structures for the DSMR Algorithm

P	$\equiv$	Queue of permanent routes to all nodes.
P_n	$\equiv$	Queue of permanent routes to node n.
T	$\equiv$	Heap of temporary routes.
T_n	$\equiv$	Entry in T for node n.
B_n	$\equiv$	Balanced tree of routes for node n.
$\mathcal{E}_n$	$\equiv$	Summary of traffic expression for all routes in P_n.
$A(i)$	$\equiv$	The set of edges adjacent to i in the graph.

TABLE I
NOTATION.

$\neg(r \sqsubseteq s)$. The maximum size of a maximal subset of routes is the smallest range of the components of the weights. The following path algebra, based on weights composed of delay and bottleneck bandwidth, implements what is commonly called *widest-shortest* routing where the path with the most bandwidth is selected from the set of paths with the least delay:

$$\begin{aligned}
\omega_i &\equiv (d_i, b_i)\\
\overline{0} &\equiv (0, \infty)\\
\overline{\infty} &\equiv (\infty, 0)\\
\omega_i \oplus \omega_j &\equiv (d_i + d_j, Min(b_i, b_j))\\
\omega_i \preceq \omega_j &\equiv (d_i < d_j) \vee ((d_i = d_j) \wedge (b_i \geq b_j))\\
\omega_i \sqsubseteq \omega_j &\equiv (d_j \leq d_i) \wedge (b_j \geq b_i)
\end{aligned}$$

Figure 4 is a graphical depiction of the relation $\preceq$ on the set of weights used as a example in Section III where $x \preceq y$ is depicted as $x \rightarrow y$. The $\sqsubseteq$ relation, illustrated by Figure 2, formalizes the *dominates* notion presented above. And, lastly, R_m formalizes the notions of performance classes in a graph, and is the best set of routes we are looking for. R_m is illustrated in Figure 3.

Figure 6 presents the DSMR algorithm, which is a modified

Notation	Description
	Queue
$Push(r, Q)$	Insert record r at tail of queue Q ($O(1)$)
$Tail(Q)$	Return record at tail of queue Q ($O(1)$)
	d-Heap
$Insert(r, H)$	Insert record r in heap H ($O(\log_d(n))$)
$IncreaseKey(r, r_h)$	Replace record r_h in heap with record r having greater key value ($O(d \log_d(n))$)
$DecreaseKey(r, r_h)$	Replace record r_h in heap with record r having lesser key value ($O(\log_d(n))$)
$Min(H)$	Return record in heap H with smallest key value ($O(1)$)
$DeleteMin(H)$	Delete record in heap H with smallest key value ($O(d \log_d(n))$)
$Delete(r_h)$	Delete record r_h from heap ($O(d \log_d(n))$)
	Balanced Tree
$Insert(r, B)$	Insert record r in tree B ($O(\log(n))$)
$Min(B)$	Return record in tree B with smallest key value ($O(\log(n))$)
$DeleteMin(B)$	Delete record in tree B with smallest key value ($O(\log(n))$)

TABLE II
OPERATIONS ON DATA STRUCTURES [18].

```
algorithm DSMR
    begin
1   $Push(<s, s, \bar{0}>,\ P_s)$;
2   for each $\{(s, j) \in A(s)\}$
3       $Insert(<j, s, \omega_{sj}>,\ T)$;
4   while $(|T| > 0)$
        begin
5       $<i, p_i, \omega_i> \leftarrow Min(T)$;
6       $DeleteMin(B_i)$;
7       if $(|B_i| = 0)$
8           then $DeleteMin(T)$
9           else $IncreaseKey(Min(B_i),\ T_i)$;
10      if $(\omega_i \not\sqsupseteq Tail(P_i).\omega)$
            then begin
11              $Push(<i, p_i, \omega_i>,\ P_i)$;
12              for each $\{(i, j) \in A(i) \mid \omega_i \oplus \omega_{ij} \not\sqsupseteq Tail(P_i).\omega\}$
                    begin
13                  $\omega_j \leftarrow \omega_i \oplus \omega_{ij}$;
14                  if $(T_j = \emptyset)$
15                      then $Insert(<j, i, \omega_j>,\ T)$
16                  else if $(\omega_j \prec T_j.\omega)$
17                      then $DecreaseKey(<j, i, \omega_j>,\ T)$;
18                  $Insert(<j, i, \omega_j>,\ B_j)$;
                    end
                end
        end
    end
```

Fig. 6. DSMR.

Dijkstra SPF algorithm that computes the maximal set of routes to each destination subject to multiple metrics. The notation used in the algorithm presented below is summarized in Table I. In addition, the maximum number of distinct performance classes is denoted by W, and the maximum number of adjacent neighbors by $a_{max} = \max\{|A(i)| \mid i \in N\}$. Table II defines the primitive operations for queues, heaps, and balanced trees used in the algorithm, and gives their time complexity used in the complexity analysis.

The algorithm presented in this section is based on the data structure model shown in Figure 5. In this structure, a balanced tree (B_i) is maintained for each node in the graph to hold newly discovered, temporary labeled routes for that node. The heap T contains the lightest weight entry from each non-empty B_i (for a maximum of n entries). A queue, P_i, is maintained for each node which contains the set of permanently labeled routes discovered by the algorithm, in the order in which they are discovered (which will be in increasing weight).

The general flow of the algorithm is to take the minimum entry from the heap T, compare it with existing routes in the appropriate P_i, if it is incomparable with existing routes in P_i it is pushed onto P_i, and "relaxed" routes for its neighbors are added to the appropriate B_x's. See [19] for a full proof of correctness.

B. Benefits of DSMR

DSMR has a number of features that make it particularly well suited to the dual challenges of QoS routing and minimizing congestion. The set of routes computed by DSMR provides the full range of performance available from a network. This provides the assurance, lacking in other proposals (e.g. [8]) that if paths exist in a network that satisfy a given flow's QoS requirements, one of them is in the set computed by DSMR.

In general, multiple routes will satisfy the QoS requirements for any given flow. This can be seen from Figure 3 in that the four paths computed by DSMR all overlap. This overlap presents the opportunity to further distribute traffic over multiple paths.

DSMR can be used with either link-state or distance-vector information. Therefore, it can be implemented as an enhancement to either link-state or distance-vector routing protocols.

Lastly, DSMR computes routes that support hop-by-hop forwarding.

IV. SIMULATIONS

This section presents simulations of the dominant set multi-path routing solution for network congestion and QoS routing. Experiments proceed by running twenty trials for a given pair of graph size and average degree. Each trial proceeds by generating a random graph with random link weights, computing both single- and multi-path routing tables for all nodes in the graph, and then processing a random stream of flow requests for 3000 seconds.

Random network topologies are generated using the GT-ITM package [20] with a target graph size and degree. Link metrics include delay and bandwidth which are uniformly distributed between 1 and 5ms for delay and 1Mbps and 1Gbps for bandwidth.

Random flows are generated with an exponential inter-arrival time with a mean defined by the target flow rate. Flows are categorized as elastic or real time, with 75% of the flows being real-time. The QoS requirements for elastic flows are 35Kbps of bandwidth, 400ms delay, and the duration of elastic flows is Poisson distributed with a mean of 30 seconds. The QoS requirements of real-time flows are 2Mbps of bandwidth and 200ms delay, and the duration of real-time flows is Poisson distributed with a mean of 120 seconds. The QoS requirements of flows were are obtained from [1]. The source and destination of a flow are randomly selected, with a uniform distribution across all nodes in the graph. The average call acceptance ratio (CAR) of the twenty trials is reported for each experiment for both single- and multii-path solutions.

Routing tables are computed once at the start of a trial using the DSMR algorithm presented above for the multi-path case. Paths are selected in a centralized manner where available bandwidth is monitored for the links in the graph and paths are selected only if they have adequate bandwidth and satisfy the QoS requirements of the flow. Call failures occur when there were no paths that both satisfy the QoS requirements of the flow and have adequate bandwidth available for the flow. The bandwidth available on each link is tracked in the simulation, and updated as accepted flows are initiated and terminated.

Given these basic CAR numbers, the imporant question is what return does DSMR routing provide for a given allocation of network resources. To answer this, Figures 7 and 8 present results showing, for each experiment, the flow rate that supports a 95% CAR. The significance of the 95% flow rate is as a measure of how congested the network is. The ideal value to use for this cutoff is the CAR at which users begin to experience degraded performance of their network applications. It is an area for future work to determine a realistic number for this measure of congestion. For this study we use it as a mechanism for measuring the relative performance of the DSMR algorithm vs. traditional single-path routing.

Figure 7(a) graphs the 95% CAR flow rate for both single-path and DSMR routing for the range of graph sizes with an average degree of 32. The important result from this graph is how much faster the capacity of the network grows with increases in graph size using DSMR as compared with single-path routing. Figure 7(b) shows the full range of results for DSMR.

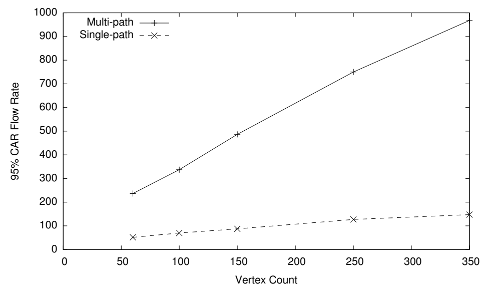
Lastly, Figure 8 graphs the ratio of DSMR to single-path routing 95% CAR flow rates for the full range of graph sizes and average degrees. This provides a measure of how much more capacity DSMR provides over single-path routing for a given network. The primary message here is how dramatically the capacity of a network improves when moving from single-path to DSMR routing, ranging from greater than a 300% improvement at the low end to greater than 1100% at the high end, with typical improvement in the range of 500% to 800%.

One interesting observation is how graphs with higher degree perform worse in this measure; from Figure 8 the optimal degree of a network, in terms of relative gain of DSMR vs. single-path routing, is in the range of 8 to 16. Note this this is only relative to single-path routing; from Figure 7(b) we can see that increases in average degree provide significant increases in capacity.

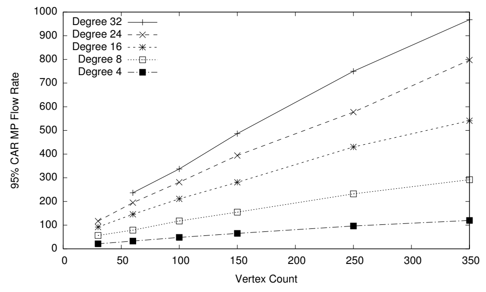
V. Conclusion

In this paper we have identified the need to address the limitations of single-path routing on congestion and QoS in the Internet. We have reviewed the datagram and hop-by-hop communication model used in the Internet, and the importance of retaining this model in a multi-path routing soltution for the Internet. We reviewed previous work in this area, identifying the absence of solutions for the use of multipath routing to address congestion and QoS that are compatible with the Internet's datagram, hop-by-hop communications model.

We then reviewed a solution based on the DSMR algorithm that satisfies all of these requirements. For this solution the metrics used in the routing computation are assumed to have multiple components such as delay, available bandwidth, jitter, etc. The metrics for the set of routes between a given source and destination are seen as a partially ordered set in this multi-dimensional metric space. The DSMR algorithm computes the dominant set of metrics for this partial ordering, representing a *best set of routes* between the source and destination that provide the full range of performance in the network. A traffic classification function is then defined for assigning new flows to paths that meet the QoS requirements of the flow and have capacity for the new flow.



(a) MP and SP @ Degree 32



(b) MP for all Degrees

Fig. 7. 95% CAR Flow Rate

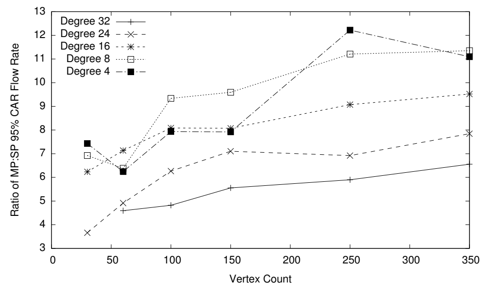


Fig. 8. Capacity Gain from Increased Network Resources

Lastly, we presented simulation results using centralized path selection where, among the paths that satisfy the QoS requirements of the flow, a path is selected that has adequate bandwidth, if such a path exists, to quantify the potential gains from this solution. The results showed significant potential gains ranging from 3 to 11-fold increases in call rates relative to single-path routing while maintaining 95% or better call acceptance ratios.

A number of next steps are needed to explore the potential of this solution. Higher fidelity, protocol-based simulations are needed to determine the CAR at which the user's experience begins to degrade. And an OpenFlow-based implementation of DSMR must be done to explore the design parameters of this architecture.

References

[1] Y. Chen, T. Farley, and N. Ye, "Qos requirements of network applications on the internet," *Information, Knowledge, Systems Management*, vol. 4, no. 1, pp. 55–76, 2004.

[2] B. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC1633, Jul. 1994.

[3] B. R. Smith and J. Garcia-Luna-Aceves, "Best effort quality-of-service," *Proceedings 17th International Conference on Computer Communications and Networks (ICCCN '08).*, August 2008.

[4] S. Blake, D. L. Black, M. A. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," RFC 2475, December 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2475.txt

[5] S. Vutukury and J. Garcia-Luna-Aceves, "A simple approximation to minimum-delay routing," in *Proceedings SIGCOMM 99*, August 1999, pp. 227–238.

[6] R. G. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE Transactions on Communications*, vol. 25, no. 1, pp. 73–85, January 1977.

[7] S. Vutukury and J. J. Garcia-Luna-Aceves, "A traffic-engineering approach based on minimum-delay routing," in *Proceedings International Conference on Computer Communications and Networks 2000*, 2000, pp. 42–47.

[8] N. Taft-Plotkin, B. Bellur, and R. Ogier, "Quality-of-service routing using maximally disjoint paths," in *Proceedings 7th International Workshop on Quality of Service (IWQoS) '99*, 1999, pp. 119–128.

[9] S. Nelakuditi and Z.-L. Zhang, "On selection of paths for multipath routing," in *Proceedings 9th International Workshop on Quality of Service (IWQoS) '01*, 2001.

[10] R. Stewart, "Stream control transmission protocol," RFC 4960, September 2007.

[11] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural guidelines for multipath tcp development," RFC 6182, March 2011.

[12] F. Paganini and E. Mallada, "A unified approach to congestion control and node-based multipath routing," *IEEE/ACM Transactions on Networking*, vol. 17, no. 5, pp. 1413–1426, October 2009.

[13] V. Bui, W. Zhu, A. Botta, and A. Pescape, "A markovian approach to multipath data transfer in overlay networks," *IEEE Transaction on Parallel and Distributed Systems*, vol. 21, no. 10, pp. 1398–1411, October 2011.

[14] L. Cruvinel and T. Vazao, "Profile-based adaptive diffserv policing with learning techniques," in *Proceedings International Conference on Computer Communications and Networks 2011*, August 2011.

[15] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus netorks," *SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, March 2008.

[16] Y. Rekhter, B. Davie, D. Katz, E. Rosen, and G. Swallow, "Cisco Systems' Tag Switching Architecture Overview," RFC2105, Feb. 1997.

[17] J. L. Sobrinho, "Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 541–550, Aug. 2002.

[18] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows – Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[19] B. R. Smith, "Efficient policy-based routing in the internet," Ph.D. dissertation, University of California, Santa Cruz, 2003.

[20] E. W. Zegura, K. Calvert, and S. Bhattacharjee, "How to Model an Internetwork," in *Proceedings INFOCOM '96*. IEEE, 1996.