Policy-Aware Connectionless Routing ⋆

Bradley R. Smith and J.J. Garcia-Luna-Aceves

University of California, Santa Cruz

Abstract. The current Internet implements *hop-by-hop* packet forwarding based entirely on globally-unique identifiers specified in packet headers, and routing tables that identify destinations with globally unique identifiers and specify the next hops to such destinations. This model is very robust; however, it supports only a single forwarding class per destination. As a result, the Internet must rely on mechanisms working "on top" of IP to support quality-of-service (QoS) or traffic engineering (TE). We present the first policy-based connectionless routing architecture and algorithms to support QoS and TE as part of the basic network-level service of the Internet. We show that policy-aware connectionless routing can be accomplished with roughly the same computational efficiency of the traditional single-path shortest-path routing approach.

1 Introduction

The current Internet architecture is built around the notion that the network layer provides a single-class best-effort service. This service is provided with routing protocols that adapt to changes in the Internet topology, and a packet forwarding method based on a single class of service for all destinations. Using one or more routing protocols, each router maintains a routing-table entry for each destination specifying the globally unique identifier for the destination (i.e., an IP address range) and the next hop along the one path chosen for the destination. Based on such routing tables, each router forwards data packets independently of other routers and based solely on the next-hop entries specified in its routing table. This routing model is very robust. However, there are many examples of network performance requirements and resource usage policies in the Internet that are not homogeneous, which requires supporting multiple service classes [2].

Policy-based routing involves the forwarding of traffic over paths that honor policies defining performance and resource-utilization requirements. *Quality-of-service* (QoS) routing is the special case of policy-based routing in the context of performance policies, and *traffic engineering* (TE) is routing in the context of resource-utilization policies. Section 2 reviews previous solutions for supporting QoS and TE. All past and current approaches to supporting QoS and TE in the Internet have been implemented "on top" of the single-class routing tables of the basic Internet routing model. This leads to inefficient allocation of the available bandwidth, given that paths computed based

⋆ This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under Grant N66001-00-8942 and by the Baskin Chair of Computer Engineering at UCSC.

on shortest-path routing within autonomous systems have little to do with QoS and TE constraints. Furthermore, current proposals for policy-based routing [1] are connection-oriented, and require source-specified forwarding implemented by source routing or some form of path setup.

There are two key reasons why policy-based path selection (with QoS and TE constraints) has not been addressed as an integral part of the basic routing model of the Internet. Routing with multiple constraints is known to be NP hard [8], and the basic Internet packet-forwarding scheme is based on globally-unique destination identifiers. This paper introduces the first policy-aware connectionless routing model for the Internet addressing these two limitations. It consists of the routing architecture presented in Section 3, and the path-selection algorithms presented in Section 4. The proposed policy-aware connectionless routing (PACR) architecture is the first to extend the notion of label swapping and threaded indices [4] into connectionless packet forwarding with multiple service classes. The path-selection algorithms we introduce generalize Dijkstra's shortest path first (SPF) algorithm to account for *both* TE and QoS constraints. These algorithms have been shown to be correct (i.e., they compute loop-less paths satisfying TE and QoS constraints within a finite time) [12], and are the first of their kind to attain computational efficiencies close to that of SPF for typical Internet topologies.

2 Previous Work

Resource Management: Two Internet QoS architectures have been developed for resource management: The *integrated services* (intserv) architecture [2], and the *differentiated services* (diffserv) architecture. Both work "on top" of an underlying packet forwarding scheme. In Intserv, network resources must be explicitly controlled; applications reserve the network resources required to implement their functionality; and admission control, traffic classification, and traffic scheduling mechanisms implement the reservations. Diffserv provides resource management without the use of explicit reservations. A set of *per-hop forwarding behaviors* (PHBs) is defined within a diffserv domain to provide resource management services appropriate to a class of application resource requirements. Traffic classifiers are deployed at the edge of a diffserv domain, which classify traffic for one of these PHBs. Inside a diffserv domain, routing is performed using the traditional hop-by-hop, single-class mechanisms.

Resource management for TE is quite simple. The desired resource utilization policies are used as constraints to the path-selection function, and traffic classification and policy-based forwarding mechanisms are used to implement the computed paths. Current proposals [1] define resource-utilization policies by assigning network resources to resource classes, and then specifying what resource classes can be used for forwarding each traffic class.

Routing Architectures: Currently proposed policy-based routing architectures are based on a centralized routing model where routes are computed on-demand (e.g. on receipt of the first packet in a flow, or on request by a network administrator), and forwarding is source-specified through the use of source routing or path setup [6] techniques. These solutions are less robust, efficient, and responsive than the original distributed routing method. The forwarding paths in on-demand routing are brittle, because

the ingress router controls remote forwarding state in routers along paths it has set up, and must re-establish the paths in the event that established paths are broken.

Path Selection: The seminal work on the problem of computing paths in the context of more than one additive metric was done by Jaffe [8], who defined the multiply-constrained path problem (MCP) as the computation of paths in the context of two additive metrics. He presented an enhanced distributed Bellman-Ford (BF) algorithm that solved this problem with time complexity of $O(n^4 b \log(nb))$, where n is the number of nodes in a graph, and b is the largest possible metric value. Many solutions have been proposed for computing exact paths in the context of multiple metrics for special situations since Jaffe's work. Wang and Crowcroft [14] were the first to present the solution to computing paths in the context of a concave (i.e. "minmax") and an additive metric. Ma and Steenkiste [9] presented a modified BF algorithm that computes paths satisfying delay, delay-jitter, and buffer space constraints in the context of weighted-fair-queuing scheduling algorithms in polynomial time. Cavendish and Gerla [3] presented a modified BF algorithm with complexity of $O(n^3)$ which computes multi-constrained paths if all metrics of paths in an internet are either non-decreasing or non-increasing as a function of the hop count. Recent work by Siachalou and Georgiadis [11] on MCP has resulted in an algorithm with complexity $O(nW \log(n))$. This algorithm is a special case of the policy-based path-selection presented in Section 4 of this paper.

Several other algorithms have been proposed for computing approximate solutions to the QoS path-selection problem. Both Jaffe [8] and Chen and Nahrstedt [5] propose algorithms which map a subset of the metrics comprising a link weight to a reduced range, and show that using such solutions, the cost of a policy-based path computation can be controlled at the expense of the accuracy of the selected paths. Similarly, a number of researchers [8, 10] have presented algorithms that compute paths based on a function of the multiple metrics comprising a link weight. These approximation solutions do not work with administrative traffic constraints.

3 Policy-Aware Connectionless Routing (PACR)

Policy-based routing requires the ability to compute and forward traffic over multiple paths for a given destination. For TE, multiple paths may exist that satisfy disjoint network usage policies. For QoS, there may not exist a universally "best" route to a given node in a graph. For example, which of two paths is best when one has delay of $5ms$ and jitter of $4ms$, and the other has delay of $10ms$ and jitter of $1ms$ depends on which metric is more critical for a given application. For FTP traffic, where delay is important and jitter is not, the former would be more desirable. Conversely, for video streaming, where jitter is very important and delay is relatively un-important, the latter would be preferred. Such weights are said to be *incomparable.* In contrast, it is possible for one route to be clearly "better than" another in the context of multi-component link weights. For instance, a route with delay of $5ms$ and jitter of $1ms$ is clearly better than a route with delay of $10ms$ and jitter of $5ms$ for all possible application requirements. Such weights are said to be *comparable.*

The goal of routing in the context of multi-component link weights is to find *the largest set of paths to each destination with weights that are mutually incomparable.*

The weights in such a set are called the *performance classes* of a destination. Supporting policy-based connectionless routing requires three main functions: (a) computing and maintaining routes that satisfy QoS and TE constraints for each destination, (b) classifying traffic intended for a given destination on the basis of TE and QoS constraints, and (c) forwarding the classified traffic solely on the basis of the next hops specified in the routing tables of routers for a given destination and for a given traffic class. The rest of this section outlines our proposed solution, which we call PACR (policy-aware connectionless routing).

3.1 Policy-Aware Route Computation

The routing protocols used in PACR must be designed to carry the link metrics required to implement the desired QoS and TE policies. This requires the use of either a topology-broadcast (also called "link-state") or link-vector routing protocol [7] that exchanges information describing the state of links. The implementation of these routing protocols consists of two main parts: (a) information exchange signaling, and (b) local path-selection.

The signaling component of the protocols is straightforward, because it suffices to re-engineer the signaling of one of many existing routing protocols to accommodate QoS and TE parameters of links. As we discuss subsequently, routing-table information can be exchanged in such signaling, in addition to link-state information. The path-selection component of the protocols is the complex part of PACR, because the path-selection algorithm used must produce paths that satisfy QoS and TE constraints at roughly the same speed with which today's shortest-path algorithms compute paths in a typical Internet topology. Section 4 presents the new path-selection algorithms required in PACR, which arguably constitute the main contribution of the new architecture.

3.2 Packet Forwarding

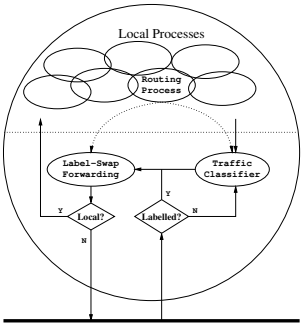

Fig. 1. Traffic flow in PACR

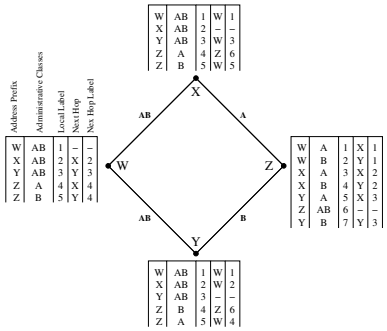

Fig. 2. Forwarding labels in PACR

Solutions for packet classification already exist and can be applied to distributed policy-based routing. However, forwarding packets solely based on IP addresses would require each relay of a packet to classify the packet before forwarding according to the content of its routing table. We propose using label-swap forwarding technology to require only the first router that handles a packet to classify it before forwarding it. Accordingly, the forwarding state of a router must be enhanced to include local and next hop label information, in addition to the destination and next hop information existing in traditional forwarding tables. Traffic classifiers must then be placed at the edge of an internet, where "edge" is defined to be any point from which traffic can be injected into the internet. Figure 1 illustrates the resulting traffic flow requirements of a router in PACR.

To date, label-swapping has been used in the context of connection-oriented (virtual circuit) packet forwarding architectures. A connection setup phase establishes the labels that routers should use to forward packets carrying such labels, and a label refers to an active source-destination connection [6]. Chandranmenon and Varghese [4] present *threaded indices*, in which neighboring routers share labels corresponding to indexes into their routing tables for routing-table entries for destinations, and such labels are included in packet headers to allow rapid forwarding-table lookups.

The forwarding labels in PACR are similar to threaded indices. A label is assigned to each routing-table entry, and each routing-table entry corresponds to a policy-based route maintained for a given destination. Consequently, for each destination, a router exchanges one or multiple labels with its neighbors. Each label assigned to a destination corresponds to the set of service classes satisfied by the route identified by the label. For example, Figure 2 shows a small network with four nodes with the forwarding tables at each node, two administrative classes A and B, and the given forwarding state for reaching the other nodes.

4 Policy-Based Path-Selection Algorithm

We model a network as a weighted undirected graph $G = (N, E)$, where N and E are the node and edge sets, respectively. By convention, the size of these sets are given by $n = \mid N \mid$ and $m = \mid E \mid$. Elements of E are unordered pairs of distinct nodes in N. $A(i)$ is the set of edges adjacent to i in the graph. Each link $(i, j) \in E$ is assigned a weight, denoted by ω_{ij}. A *path* is a sequence of nodes $< x_1, x_2, \ldots, x_d >$ such that $(x_i, x_{i+1}) \in E$ for every $i = 1, 2, \ldots, d-1$, and all nodes in the path are distinct. The weight of a path is given by $\omega_p = \sum_{i=1}^{d-1} \omega_{x_i x_{i+1}}$. The nature of these weights, and the functions used to combine these link weights into path weights are specified for each algorithm.

4.1 Administrative Policies

We use a *declarative* model of administrative policies in which constraints on the traffic allowed in an internet are specified by expressions in a boolean traffic algebra. The *traffic algebra* is composed of the standard boolean operations on the set $\{0, 1\}$, where a set of p primitive propositions (variables) represent statements describing characteristics of

network traffic or global state that are either true or false. The syntax for expressions in the algebra is specified by the BNF grammar:

$$\varphi ::= 0 \mid 1 \mid v_1 \ldots v_p \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi)$$

The set of primitive propositions, indicated by v_i in the grammar, can be defined in terms of network traffic characteristics or global state. Administrative policies are specified for an internet by assigning expressions in the algebra to links in the graph, called *link predicates*. These predicates define a set of *forwarding classes*, and constrain the topology that traffic for each forwarding class is authorized to traverse, as required by the administrative policies.

A $SAT(\varphi)$ primitive is required for expressions in the traffic algebra which is the SATISFIABILITY problem of traditional propositional logic. Satisfiability must be tested in two situations: to determine if traffic classes exist that are authorized to use an extension to a known route, and to determine if all traffic authorized for a new route is already satsified by known shorter routes. The first is true *iff* the conjunction of these expressions is satisfiable (i.e., $SAT(\varepsilon_i \wedge \varepsilon_{ij})$). The second is true *iff* the new route's traffic expression implies the disjunction of the traffic expressions for all known better routes (i.e., $(\varepsilon_i \rightarrow \varepsilon_{i_1} \vee \varepsilon_{i_2} \vee \ ..)$ is *valid*, which is denoted by $(\varepsilon_i \rightarrow \mathcal{E}_i)$ in the algorithms). Determining if an expression is valid is equivalent to determining if the negation of the expression is unsatisfiable. Therefore, expressions of the form $\varepsilon_1 \rightarrow \varepsilon_2$ are equivalent to $\neg SAT(\neg(\varepsilon_1 \rightarrow \varepsilon_2))$ (or $\neg SAT(\varepsilon_1 \wedge \neg\varepsilon_2)$). Satisfiability has many restricted versions that are computable in polynomial time. We have implemented an efficient, restricted solution to the SAT problem by implementing the traffic algebra as a set algebra with the set operations of intersection, union, and complement on the set of all possible forwarding classes.

4.2 Performance Characteristics

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. Our path-selection algorithm is based on an enhanced version of the path algebra defined by Sobrinho [13], which we enhance to support the computation of the best *set* of routes for each destination. Formally, the path algebra $P = < \mathcal{W}, \oplus, \preceq, \sqsubseteq, \overline{0}, \overline{\infty} >$ is defined as a set of weights $\mathcal{W}$, with a binary operator $\oplus$, and two order relations, $\preceq$ and $\sqsubseteq$, defined on $\mathcal{W}$. There are two distinguished weights in $\mathcal{W}$, $\overline{0}$ and $\overline{\infty}$, representing the least and absorptive elements of $\mathcal{W}$, respectively. Operator $\oplus$ is the original path composition operator, and relation $\preceq$ is the original total ordering from [13]. Operator $\oplus$ is used to compute path weights from link weights. The relation $\preceq$ is used by the routing algorithm 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.

We add a new relation on routes, $\sqsubseteq$, to the algebra and use it to define classes of comparable routes to 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 additional property that $(\omega_x \sqsubseteq \omega_y) \Rightarrow (\omega_x \succeq$

```
algorithm Policy-Based-Dijkstra
    begin
1   Push(<s, s, 0̄, 1>, P_s);
2   for each {(s, j) ∈ A(s)}
3     Insert(<j, s, ω_sj, ε_sj>, T);
4   while (|T| = 0)
      begin
5     <i, p_i, ω_i, ε_i> ← Min(T);
6     DeleteMin(B_i);
7     if (|B_i| = 0)
8       then DeleteMin(T)
9       else IncreaseKey(Min(B_i), T_i);
10    ε_tmp ← ε_i; ptr ← Tail(P_i);
11    while ((ε_tmp ≠ 0) ∧ (ptr ≠ ∅))
12      ε_tmp ← ε_tmp ∧ ¬ptr.ε; ptr ← ptr.next;
13    if (ε_tmp ≠ 0)
        then begin
14      Push(<i, p_i, ω_i, ε_i>, P_i);
15      for each {(i, j) ∈ A(i) | SAT(ε_i ∧ ε_ij)}
          begin
16        ω_j ← ω_i ⊕ ω_ij; ε_j ← ε_ij;
17        if (T_j = ∅)
18          then Insert(<j, i, ω_j, ε_j>, T)
19        else if (ω_j ≺ T_j.ω)
20          then DecreaseKey(<j, i, ω_j, ε_j>, T);
21        Insert(<j, i, ω_j, ε_j>, B_j);
          end
        end
    end
end
```

Fig. 3. General-Policy-Based Dijkstra.

ω_y). The relation $\sqsubseteq$ defines an ordering on routes in terms of the containment (subset) of the set of constraints satisfied by one route within the set satisfied by another, i.e., if $\omega_i \sqsubseteq \omega_j$, then the set of constraints that route i can satisfy is a subset of those satisfiable by route j.

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 $s \in R - \{r\}$, $\neg(r \sqsubseteq s)$. The maximum size of a maximal subset of routes is the smallest range of the components of the weights (for the two component weights considered here).

4.3 Path Selection

Path selection in PACR consists of computing the maximal set of routes to each destination in an internet for each traffic class (stated through link predicates and multi-component link weights) for which a path to the destination exists.

The path-selection algorithm in PACR maintains a balanced tree (B_i) for each node i in the graph to hold newly discovered, temporary labeled routes for node i. A heap T contains the lightest weight entry from each non-empty B_i (for a maximum of n entries), and the heap entry for node i is denoted by T_i. Lastly, 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 path-selection 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 add "relaxed" routes for its neighbors to the appropriate B_x's.

The correctness of the PACR path-selection algorithm is based on the maintenance of the following three invariants: for all routes $I \in P$ and $J \in B_*$, $I \preceq J$, all routes to a given destination i in P are incomparable for some set of satisfying truth assignments, and the maximal subset of routes to a given destination j in $P_j \cup B_j$ represents the maximal subset of all paths to j using nodes with routes in P. Furthermore,

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.

Table 1. Notation.

Notation	Description
	Queue
$Push(r, Q)$	INSERT RECORD r AT TAIL OF QUEUE Q ($O(1)$)
$Head(Q)$	RETURN RECORD AT HEAD OF QUEUE Q ($O(1)$)
$Pop(Q)$	DELETE RECORD AT HEAD OF QUEUE Q ($O(1)$)
$PopTail(Q)$	DELETE 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 SMALLER 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 2. Operations on data structures.

these invariants are maintained by the following two constraints on actions performed in each iteration of these algorithms: (1) only known-non-maximal routes are deleted or discarded, and (2) only the smallest known-maximal route to a destination i is moved to P_i. The details of this proof are presented elsewhere [12].

The PACR path-selection algorithm, presented in Figure 3 computes an optimal set of routes to each destination subject to multiple general (additive or concave) path metrics, in the presence of traffic constraints on the links. The notation used in the algorithms presented in the following is summarized in Table 1. Table 2 defines the primitive operations for queues, heaps, and balanced trees used in the algorithms, and gives their time complexity used in the following analysis. The worst-cast time complexity of Policy-Based-Dijkstra is $O(nW^2A^2)$, where the maximum number of unique truth assignments is denoted by $A = 2^p$ (p is the number of primitive propositions in the traffic algebra), and the maximum number of unique weights by $W =$ min(*range of weight components*). The performance of special-case variants of this algorithm for traffic-engineering and QoS (called the "Basic" algorithms below) are $O(mA\log(A))$ and $O(mW\log(W))$, respectively. Furthermore, for these variants, refinements in the data structures result in algorithms (called the "Enhanced" algorithms below) with $O(mA\log(n))$ and $O(mW\log(n))$ complexity. Details of these variants and the complexity analysis are presented elsewhere [12].

4.4 Performance Results

Figures 4 and 5 present performance results for the path-selection algorithm. The experiments were run on a 1GHz Intel Pentium 3 based system. The algorithms were implemented using the C++ Standard Template Library (STL) and the Boost Graph Library. Each test involved running the algorithm on ten random weight assignments to ten randomly generated graphs (generated using the GT-ITM package [15]).

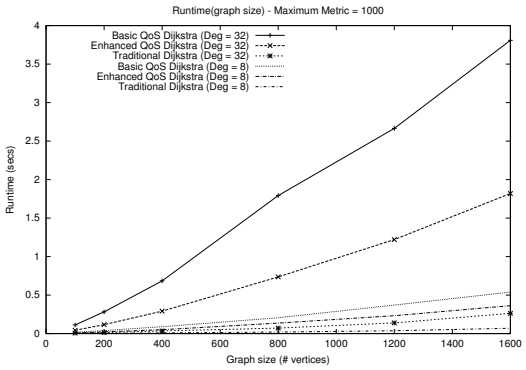

Fig. 4. QoS Runtime(Size)

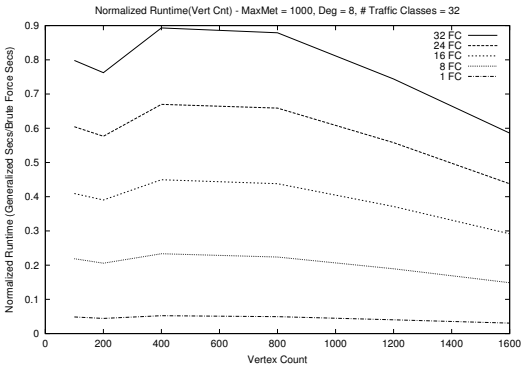

Fig. 5. TE Norm Runtime(Size)

Fig. 4 show the worst-case measurements for each test of the QoS algorithms. The "Traditional" algorithm is an implementation of Dijkstra's shortest-path-first (SPF) algorithm using the same environment as that used for the other algorithms for use as a reference. The metrics were generated using the "Cost 2" scheme from [11], where the delay component is randomly selected in the range $1..MaxMetric$, and the cost component is computed as $cost = \sigma(MaxMetric - delay)$, where σ is a random integer in the range $1..5$; this scheme was chosen as it proved to result in the most challenging computations from a number of different schemes considered.

Tests were run for performance (both runtime and space) as a function of graph size, average degree of the graph, and the maximum link metric value. Due to space constraints, only the graphs for runtime as a function of size are shown here with a maximum metric of 1000. These results show that, while costs increase with both graph size and average degree, the magnitude and rate of growth are surprisingly tame for what are fundamentally non-polynomial algorithms.

Fig. 5 shows the performance of the "Basic" traffic engineering algorithm on a similar range of parameters. Each data point represents the worst performance of the algorithm out of 9 runs (3 randomly generated graphs with 3 random link weight assignments each). To control the number of forwarding classes in a graph, each graph was generated as two connected subgraphs. *Bridge* links were then added between 32 randomly selected pairs of vertices from each subgraph to form a single graph with at most 32 paths between any two nodes in different original subgraphs. 32 tests of the algorithm are then run with all traffic classes initially allocated to one bridge link (resulting in one forwarding class for all 32 traffic classes), and successive runs are performed with traffic classes distributed over one additional link for each test, with the final run allowing one traffic class over each bridge link (resulting in a one-to-one mapping of traffic classes to forwarding classes). In each test, the link predicate of all non-bridge links is set to allow all traffic classes (i.e. it is set to *true*). Each plot shows the results for 1, 8, 16, 24, and 32 forwarding classes in terms of the runtime of the algorithm normalized as a fraction of the "Brute Force" runtime required to run the traditional Dijkstra algorithm once for each traffic class. The plots show that the algo-

rithm provides significant savings when the number of forwarding classes is small, and gracefully degrades as the number of forwarding classes grows.

5 Conclusions

We have defined policy-aware routing as the computation of paths, and the establishment of forwarding state to implement paths, in the context of non-homogeneous performance requirements and network usage policies. We showed that a fundamental requirement of policy-aware routing is support for multiple paths to a given destination, and that the address-based, single-forwarding-class Internet routing model cannot support such a requirement. We presented PACR, which is the first policy-based connectionless routing architecture, and includes the first policy-based routing solution that provides integrated support of QoS and TE. The path-selection algorithms introduced for PACR constitute the most efficient algorithms for path selection with QoS and TE constraints known to date. Furthermore, their computational efficiency is comparable to that of shortest-path routing algorithms, which makes policy-aware connectionless routing in the Internet feasible.

References

1. D. O. Awduche, J. Malcom, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for Traffic Engineering Over MPLS," RFC 2702, December 1999.
2. B. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633, June 1994.
3. D. Cavendish and M. Gerla. "Internet QoS Routing using the Bellman-Ford Algorithm," *Proc. IFIP Conference on High Performance Networking*, 1998.
4. G. P. Chandranmenon and G. Varghese, "Trading Packet Headers for Packet Processing," *IEEE ACM Trans. Networking*, 4(2):141–152, Oct. 1995.
5. S. Chen and K. Nahrstedt, "An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions," *IEEE Network*, pp. 64–79, Nov. 1998.
6. B. Davie and Y. Rekhter, *MPLS: Technology and Applications*, Morgan Kaufmann, 2000.
7. J.J. Garcia-Luna-Aceves and J. Behrens, "Distributed, Scalable Routing Based on Vectors of Link States," *IEEE JSAC*, Oct. 1995.
8. J. M. Jaffe, "Algorithms for Finding Paths with Multiple Constraints," *Networks*, 14(1):95–116, 1984.
9. Q. Ma and P. Steenkiste, "Quality-of-Service Routing for Traffic with Performance Guarantees," *Proc. 4th International IFIP Workshop on QoS*, May 1997.
10. P. Van Mieghem, H. De Neve, and F. Kuipers, "Hop-by-hop quality of service routing," *Computer Networks*, 37:407–423, November 2001.
11. S. Siachalou and L. Georgiadis, "Efficient QoS Routing," *Proc. Infocom'03*, April 2003.
12. Brad Smith, " Efficient Policy-Based Routing in the Internet ", PhD Thesis, Computer Science, University of California, Santa Cruz, CA 95064, September 2003.
13. J. L. Sobrinho, "Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet," *IEEE/ACM Trans. Networking*, 10(4):541–550, Aug. 2002.
14. Z. Wang and J. Crowcroft, "Quality-of-Service Routing for Supporting Multimedia Applications," *IEEE JSAC*, pp. 1228–1234, Sept. 1996.
15. E. W. Zegura, K. Calvert, and S. Bhattacharjee, "How to Model an Internetwork," *Proc. IEEE Infocom '96*, 1996.