

# Unicast Routing: Cost-Performance Tradeoffs \*

SELMA YILMAZ

IBRAHIM MATTA

*Computer Science Department  
Boston University  
Boston, MA 02215, USA*

{selma,matta}@cs.bu.edu

Technical Report BUCS-TR-2002-018

## Abstract

*The objective of unicast routing is to find a path from a source to a destination. Conventional routing has been used mainly to provide connectivity. It lacks the ability to provide any kind of service guarantees and smart usage of network resources. Improving performance is possible by being aware of both traffic characteristics and current available resources. This paper surveys a range of routing solutions, which can be categorized depending on the degree of the awareness of the algorithm: (1) QoS/Constraint-based routing solutions are aware of traffic requirements of individual connection requests; (2) Traffic-aware routing solutions assume knowledge of the location of communicating ingress-egress pairs and possibly the traffic demands among them; (3) Routing solutions that are both QoS-aware as (1) and traffic-aware as (2); (4) Best-effort solutions are oblivious to both traffic and QoS requirements, but are adaptive only to current resource availability. The best performance can be achieved by having all possible knowledge so that while finding a path for an individual flow, one can make a smart choice among feasible paths to increase the chances of supporting future requests. However, this usually comes at the cost of increased complexity and decreased scalability. In this paper, we discuss such cost-performance tradeoffs by surveying proposed heuristic solutions and hybrid approaches.*

## 1. Introduction

The primary function of unicast routing is to find a path from a source to a destination. While finding a path, the routing algorithm should try to find an optimal route, not to misroute the packets or create loops, and not to cause oscillations. Another important requirement is scalability: With the increasing size of the network, space/communication/time complexities should increase at

a much slower rate. The main cost of routing originates from the size of the routing tables, the processing requirement of update packets, and route calculations. Conventional IP routing is shortest-path, destination-based only routing, which mainly uses static link metrics like hop count. With static routing, since metrics change only when topology changes, it is very stable. It is also connectionless and therefore cannot provide any kind of service guarantees. Since forwarding tables keep a single state per destination, conventional IP routing is highly scalable. Scalability is further improved by means of aggregating information through hierarchical addressing and routing. However, by only being aware of topology, it cannot provide any mechanisms to enable smart usage of resources in the network. The links along the shortest paths may get overutilized while alternate paths stay under-utilized. Better utilization of resources can be achieved by making use of alternate paths. Savage *et al.* [15] show that in 30-80% of the cases there is an alternate path with significantly superior quality, where quality is measured in terms of loss rate, bandwidth and round trip time (RTT).

In this context, a range of routing solutions have been proposed to increase the utilization of the network. These solutions can be categorized depending on the additional information available to the algorithm: (1) QoS/Constraint-based routing solutions are aware of traffic requirements of individual connection requests; (2) Traffic-aware routing solutions assume knowledge of the location of communicating ingress-egress pairs and possibly the traffic demands among them; (3) Routing solutions that are both QoS-aware as (1) and traffic-aware as (2); (4) Best-effort solutions are oblivious to both traffic and QoS requirements, but are adaptive only to current resource availability.

The minimal requirement for efficient usage of resources is to be aware of the available resources. Therefore, most of the solutions are dynamic routing algorithms, where the link metric is a function of some congestion measure such as delay or packet loss. Link state updates are distributed and routes are recalculated for the new states.

The taxonomy of the solutions covered in this paper can

---

\*This work was supported in part by NSF grant ANI-0095988.

be seen in Figure 1. We will discuss how increased com-

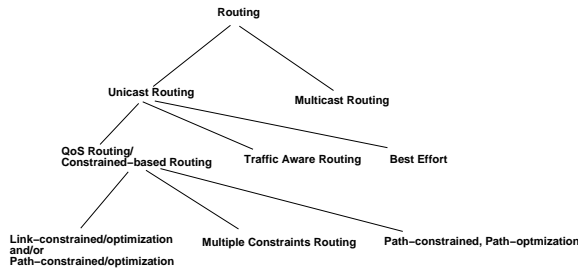


Figure 1. Taxonomy of Solutions

plexity can help improve performance, and what are the drawbacks/limitations of the solutions.

The rest of the paper is organized as follows: Section 2 gives a brief review of algorithms that fall into the category of best effort solutions. Section 3 describes the basic functionality and difficulties of QoS routing and associated cost-performance tradeoffs. Section 4 overviews how performance can be improved if the location of ingress-egress pairs and/or the traffic matrix are known. Section 5 presents simulation results comparing algorithms that fall into different categories. Section 6 concludes the paper with future research directions.

## 2 Best-effort Solutions

The algorithms falling in this category are both aware of the topology and available resources, which is one step improvement over conventional routing. The main algorithms that can be listed as best-effort are per-packet dynamic routing and load balancing along equal length paths. **Per-packet dynamic routing** is a traffic engineering tool, with which it is possible to avoid congested links if the link cost defines the way in which traffic load is distributed. It is computationally very simple; only requires Dijkstra or Bellman-Ford-like algorithm for shortest path computation. Per-packet dynamic routing is stateless, therefore scalable. The main difficulty with per-packet dynamic routing is link states change at the packet level. As the experience with ARPANET showed [8], packet level granularity for dynamic routing is too fine. It is impractical to generate link state updates at the packet level while larger link state update periods may cause oscillations because of stale link states. The oscillations maybe limited by choosing an appropriate link metric: The metric should be able to predict future loads on the links. For example, if the goal is to minimize individual packet delays, the metric should be some function of queuing delay. Using averages instead of instantaneous queue size helps, but the propagation and transmission components of delay should also be included. The route oscillations can be reduced by limiting the variations reported on successive updates for a link. The third version of ARPANET [8] suggests to use metric values that are rel-

ative to the cost of alternate links (hop-normalized). This way, under heavy load, all links will be desirable to some degree.

**Load balancing along equal length paths** has been proposed to remedy the load balancing inability of static routing. The idea is to distribute traffic equally along equal cost shortest paths. This can be achieved by using (1) per-packet round robin, or (2) source-destination address based hashing. The first approach is not advised since path characteristics along the different shortest paths may be different. If this is the case, packets for the same source-destination pair may arrive at the destination out of order and the performance of TCP-like protocols will suffer. Retransmissions triggered by out-of-order delivery will just waste bandwidth. Paxson [13] suggests avoiding packet level load balancing, especially for inter-AS load balancing. Since the granularity of source-destination address based hashing is coarser (flow level), it avoids the problems caused by per-packet round robin. However, its ability to balance load is highly affected by the availability of flows with different source-destination pairs. OSPF-ECMP is an example of protocol that uses load balancing along equal length paths. The approach is static routing if weights are administratively assigned. For example, Cisco suggests using  $1/\text{link capacity}$  as a link weight (cost). This way of balancing load may not be efficient, because it is not aware of the actual link loads on the equal cost paths. However, Lorenz *et al.* [9] show that conventional IP routing is  $\Omega(N)$  worse than OSPF-ECMP routing, where  $N$  is the number of flows and the metric is throughput (or maximum utilization).

With dynamic routing, we can achieve better distribution of traffic. However, dynamic routing introduces a new problem, which is stability. Stability is determined by observing how often the routes change. If the network is unstable, routers spend too much time updating their routing tables, and propagating the changes. With each change of link state, there is a potential of changing the current route to a destination for a better path. This is not the case for static routing: A path to a destination never changes unless topology changes. To improve stability, it is suggested that the routing protocol should not target only the best routes. If many sources pick the same best path, the best path will get overloaded and an alternate path will stay under-utilized. The current best path and alternate path can then keep changing roles at each link state update period. An example of not targeting the single best path is by using a hop-normalized link metric, and another one is by quantizing link metric values. The latter helps to improve stability by increasing the number of equal cost target paths as possible link metric values become limited. Similarly, policy-based routing suffers from stability problems. With BGP, each domain is allowed to formulate independently its routing policies and can override distance metrics in favor of policy constraints. With each route advertisement, there is a potential for a domain to change the best route so it suits its policy better. It is shown that policy-based rout-

ing may diverge, and result in oscillations in the global Internet. Therefore, extending policy-based routing for QoS, which requires dynamic routing with its own stability issues, is more challenging. That is why the research so far concentrated on intra-domain dynamic/QoS routing.

### 3 QoS/Constrained-based Routing Solutions

The goal of QoS routing is to identify paths that have sufficient resources to satisfy a set of constraints, where typical constraints are bandwidth, maximum delay, cost, etc. QoS routing algorithms should be aware of topology, available resources, and traffic requirements of the individual demands, so that it will be possible to find feasible paths effectively. QoS routing is a radical shift from the traditional routing approach: To be able to provide guarantees for end-to-end performance, after finding feasible paths, resources are reserved so that the admitted flow will be immune from the traffic variations along the links that it is using. This process is called *route pinning*. By means of admission control, if a demand cannot be guaranteed, it is not accepted at all.

QoS routing also aims to improve long term utilization of the network. Admission control can be used for this purpose. If accepting a request would put the network in a state of high blocking probability, the request should simply be rejected. QoS routing prefers social paths, which are the paths that using them does not significantly decrease the probability of accepting future requests. Similarly, efficient usage of network resources also improves long term utilization such as preferring shortest (min-hop) paths to limit resource consumption and/or preferring least loaded paths to balance load.

However, new functionalities that QoS routing promises come with a number of new challenges, which can be listed as follows:

- Performance depends on the accuracy of the network state information which changes frequently. Keeping up-to-date network state information is expensive because of flooding, which consumes bandwidth and processing power. Therefore, there is a tradeoff between communication/processing overhead and accuracy.
- QoS routing requires more sophisticated route computations to satisfy multiple constraints, and some combination of constraints makes the problem NP-hard. The tradeoff is between simpler path computations and better quality of paths.
- QoS routing requires more frequent route computation, maybe on-demand whenever a flow request arrives. The tradeoff is between per-request computational cost and the quality of the path returned.
- With QoS routing, paths need to be pinned and maintained afterwards, which introduces per-flow state. Depending on the granularity of a flow, there maybe scalability problems. If the granularity is fine, the routing

tables will be larger, lookups will be slower, and signaling cost to set-up paths, tear down and keep paths alive will grow.

Some ways to deal with the increased cost of QoS routing can be listed as follows:

- The volume of updates can be reduced by controlling how often the updates are sent. Updates can be sent periodically, or triggered by a change bigger than a specified value (threshold/class-based triggers), or clamp-down timers<sup>1</sup>. The sensitivity level of the triggering policy affects the volume of updates: While a zero clamp-down timer along with a sensitive triggering policy (e.g. small threshold) will lead to high cost, a large clamp down timer will lead to low cost, but possibly inaccurate routing decisions.
- The volume of updates can also be controlled by choosing what to advertise. Only the link of a node that triggered the update or all the links of the node may be advertised. The tradeoff is between eliminating the need for future updates and the extra processing overhead when there is little change in the state of the other links. The state of the links can be expressed as exact or quantized values from a fixed set. The tradeoff is between reduced accuracy and increased number of equal cost paths.
- Polynomial-time heuristics can be used for simpler route computations.
- Per-request processing overhead can be reduced using pre-computation or path caching instead of on-demand.
- To reduce the amount of state maintained, and to improve scalability, coarser granularity, like class-based, can be used instead of per-flow state.
- Hybrid routing can be used in a way that QoS routing is restricted only to a group of flows and static routing is used for the rest (majority) of the flows.

#### 3.1 Closer Look at Some QoS Cost-Performance Tradeoffs

- **How often paths are computed?** The three ways of computing paths are on-demand, pre-computation, and path-caching. With **on-demand**, path computation is performed at each flow request arrival. Advantages of the method can be listed as: (1) it can yield better routes since computation uses the most recent link metrics available; (2) exact QoS requirements and destination are known at the time of path computation; (3) there is no

<sup>1</sup>Clamp-down timers are used to enforce minimum spacing between two consecutive updates.

need to store the paths. Disadvantages of on-demand routing can be listed as: (1) per-request processing overhead is high; (2) if a large clamp-down timer is used, it keeps re-discovering the same paths.

With **path pre-computation**, paths are computed either periodically or after receiving a certain number of updates. An advantage of this approach is reducing the per-request processing overhead. Disadvantages are: (1) all possible paths to all destinations for all possible QoS requirements must be pre-computed; (2) pre-computed paths must be stored; (3) it introduces a path lookup cost, which is the cost of selecting a path from the pre-computed table for a given destination and QoS requirement; (4) some paths that have been pre-computed and stored may never be used; and (5) periodic pre-computation may result in routing performance loss.

A hybrid of on-demand and path pre-computation is **path caching**. The idea is to re-use the  $k$  previously computed paths to a destination. Path caching provides a good balance between per-request processing overhead and quality of paths. With bigger cache sizes, and more entries in the cache, the ability to balance load increases. However, smaller cache sizes are preferred for faster lookup and less storage overhead. The drawback of path-caching is the need to maintain the entries. With changing link states, the bottleneck bandwidth of the cached paths get stale and need to be updated. For small update periods, the cost of path caching can exceed the cost of on-demand. Also, it introduces storage overhead at each node per-destination.

- **Type of Computation:** Depending on the location of path computation, we may have three types of computation: Hop-by-hop, source routing, and crank-back.

With **hop-by-hop**, a path is computed in a distributed manner at each node. The advantage of this approach is that the route computation is distributed, which speeds up the establishment of a path by avoiding the complete route computation at the source. A disadvantage is that if the routing tables are inconsistent, it may create loops.

The other alternative is **source routing**, in which the complete path is computed at the source and included in the header of the packet. Advantages are: (1) it is guaranteed to be loop free, since the path computed at the source is enforced; (2) it allows sophisticated route computations. Disadvantages are: (1) computation is centralized; (2) global state needs to be maintained at each node; (3) since the path is included in the header of the packet, each router has to process the header to obtain the next-hop.

A hybrid of hop-by-hop and source routing is **crank-back** as used in PNNI: Source routing is used up to the point of failure during the path set-up phase. After this point, hop-by-hop routing is used. While this ap-

proach provides a good way of dealing with imperfect source routes that result from inaccurate network state information, it increases the set up time of a path for an incoming flow.

- **Granularity:** Granularity can be either **per-flow** or **larger aggregates of multiple flows**, like class-based or destination-based.

With finer granularity, it is possible to balance load better, which leads to better long term performance and better service guarantees for individual flows. However, with a high number of flows, path computation occurs more frequently, and network state changes more frequently (at per-flow level). That is why a smaller update period is required to be able to maintain the level of accuracy required for good enough performance. In conclusion, finer granularity is expensive and because of the need to maintain per-flow state, it does not scale well.

With increasing granularity, the cost of per-flow state reduces, thus scalability increases. However, coarser granularity means higher bandwidth requirements for aggregates. High bandwidth flows are harder to route because of bandwidth fragmentation as observed in [11, 10]. Also, high bandwidth aggregates restrict the ability to balance load, and may lead to inefficient usage of resources.

- **Absolute vs. Quantized Values:** The state of the links can be expressed as exact or quantized values from a fixed range. Using quantized values increases the number of equal cost paths [1], which increases the opportunity to balance load, and to increase stability by not targeting only single best choices. Quantized values are also helpful in reducing the size of pre-computed path tables [5]. Although using quantized values may reduce the accuracy of link state information, especially if only a few quantized values are being used, the increased number of equal cost paths alleviates the effects by preventing routing mistakes over single bad choices [1].

- **Hierarchical vs. Flat:** With flat (link-state) routing, nodes have to have the full knowledge of topology, which does not scale. The cost of communication, computation, and storage is huge for large topologies. With hierarchical routing, such as PNNI or OSPF, the size of the network is reduced by aggregating the topology. Each node only needs to know the complete topology in its own group, and summarized information of the other groups. Since the amount of information that is stored and distributed is reduced, scalability improves. A drawback of this approach is the loss of detailed information as the state of logical nodes and links summarize many lower-level nodes and links. Also, as the states are aggregated, inaccuracies are also aggregated, which directly affects the quality of the se-

lected paths. In conclusion, there is a tradeoff between the amount of aggregation and accuracy.

- **Hybrid Routing:** To overcome the difficulties experienced with dynamic routing and alleviate the inefficiencies of static routing, hybrid routing can be used. The idea is to classify flows according to their characteristics and to route some class of flows dynamically while routing the rest statically. An example of this approach is proposed in [16], which suggests using dynamic per-flow routing for only long-lived flows while routing the class of short-lived flows statically. This approach effectively reduces the number of flows that are dynamically routed, which leads to less frequent route computation, and smaller per-flow state. Another advantage is that since link states change more slowly, at per-long-lived-flow level, we can afford to use larger update periods. A disadvantage is that under accurate link state information, performance can be worse than pure dynamic routing because short-lived flows are routed along static paths.
- **Link State, Distance Vector, Path Vector:** With link state routing, each router knows the entire topology. OSPF and IS-IS are examples of protocols that use link state routing. The main drawback is the need to maintain the entire topology database at each node, which does not scale well. However, having the entire view of the network, more sophisticated path selection algorithms can be used for route computation. With distance vector routing, each node only keeps a shortest path tree rooted at itself to all destinations. RIP uses this approach. With distance vector routing, routers need less storage, which scales better. The drawback of distance vector is that it cannot support sophisticated path computations and does not easily allow the computation of routes specific to the requirements of individual flows. Path vector is basically distance vector routing where routing tables keep also the corresponding path. BGP is an example protocol that uses the path vector approach. The main functionality of adding paths to the routing tables is to eliminate loops. However, this approach increases the storage requirement.

### 3.2 Effects of Topology and Traffic Characteristics on QoS Routing

Topology determines the number of candidate paths between each pair of routers. With increasing number of candidate paths, there is a better chance of balancing load and more room for performance improvement. Traffic characteristics also affect the performance gain that can be achieved by QoS routing. Under uniform load, with accurate link state information, QoS routing may perform worse than static routing. The reason is that QoS routing may cause excessive alternate routing, where longer paths with

extra resources are used. Later, these paths interfere with minimum hop traffic competing for the same links [11, 1]. For such cases, trunk reservation is suggested so that capacity used by alternate routing is limited and direct traffic will always be able to find room. Under non-uniform load, QoS routing effectively improves performance by avoiding hot spots.

### 3.3 QoS Routing Problems

QoS routing problems can be classified according to the type of QoS metric that is being used. QoS metrics can be additive, multiplicative, or concave. If  $m(i, j)$  is the metric for link  $(i, j)$ , then for path  $P = (s, j, \dots, x, d)$ , the path metric  $m(P)$  is

- **additive** if it is defined as  $m(P) = m(s, j) + \dots + m(x, d)$ . Examples include delay and cost.
- **multiplicative** if it is defined as  $m(P) = m(s, j) \times \dots \times m(x, d)$ . Examples include probability of successful transmission.
- **concave** if it is defined as  $m(P) = \min$  or  $\max\{m(s, j), \dots, m(x, d)\}$ . Examples include bandwidth.

In [2], the basic QoS routing problems are classified into four classes depending on the metric. If the path metric should be optimized, the problem is called **link optimization** for a concave metric and **path optimization** for an additive/multiplicative metric. Similarly, if the path metric should be constrained, the problem is called **link constrained** for a concave metric and **path constrained** for an additive/multiplicative metric. Possible composite routing problems can be obtained from these four basic QoS routing problems. Examples are link-constrained link-optimization, link-constrained path-constrained, path-constrained path-optimization, etc. The composite QoS routing problem involving two or more additive/multiplicative metrics is NP-complete, whereas the rest is solvable in polynomial time. More specifically, path-constrained path-optimization and multi-path-constrained (multiple constraints) routing problems are NP-complete, assuming QoS metrics are independent and allowed to be real numbers or unbounded integers. If all metrics are dependent on a common metric, the problem may be solvable in polynomial time. For example, if rate-based scheduling is used (like WFQ), a delay constraint can be converted to a bandwidth constraint. If all metrics except one take bounded integer values, then the problem is solvable in polynomial time using extended shortest path algorithms (e.g. Dijkstra's). In the next sections we will look more closely into these two types of NP-complete problems, for which a large volume of heuristic solutions have been proposed.

#### 3.3.1 Multiple Constraints Routing Problem (MCP)

In a multiple constraints routing problem, network graph

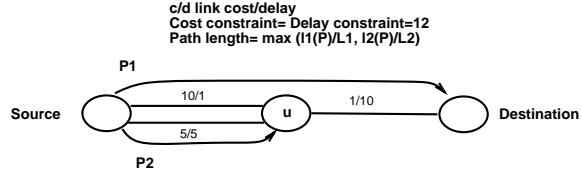
$G = (V, E)$ , with  $V$  nodes and  $E$  edges, each link connecting two nodes  $u$  and  $v$  is specified by a link vector  $l(u, v)$  having  $m$  additive link metrics  $l_i(u, v) \geq 0$  for  $i = 1, \dots, m$ . A path  $P(s, k, \dots, x, d)$  connecting a source node  $s$  and a destination node  $d$  is specified by a path vector  $l(P)$  where each component is defined as  $l_i(P) = l_i(s, k) + \dots + l_i(x, d)$ . Given  $m$  positive path constraints,  $L$ , the objective of MCP is to find a path,  $P(s, \dots, d)$  that satisfies the constraints such that  $l_i(P) \leq L_i$  for all  $i = 1, 2, \dots, m$ . The obvious question is how to run a shortest path algorithm with an  $m$ -component metric. One heuristic approach is to condense the  $m$ -component metric into a single metric (weight) by using a linear combination of the components such as  $l(P) = c_1 l_1(P) + \dots + c_m l_m(P)$ . This way, it becomes possible to use a Dijkstra-like algorithm to find the shortest path from the source to the destination. The problem with this approach can be seen from Figure 2(a): The optimal path according to the new weight function maybe infeasible.



**Figure 2. Distribution of paths in the  $(l_1(P), l_2(P))$  plane (a) with a linear path length representation, (b) with a non-linear path length representation.**

A non-linear representation of path length has been proposed to overcome this problem [12]. As can be seen from Figure 2(b), the constraint area can be scanned better when a non-linear representation of path length is used. However, with this approach, a new problem is introduced: Subsections of shortest paths are not necessarily the shortest paths any more. Therefore, Dijkstra’s algorithm fails to find shortest paths. An example of this is shown in Figure 3.

This problem of missing the shortest (feasible) path can be solved by using a  $k$ -shortest path algorithm, where at each step of Dijkstra,  $k$  shortest paths are stored at each node. As the value of  $k$  increases, the probability that a sub-path from the source to an intermediate node will be part of the shortest path from the source to the destination also increases. However, with larger  $k$  values, both run time and space complexity increase. Therefore, the value of  $k$  determines the performance and overhead of this approach.



**Figure 3.** From source to intermediate node  $u$ , Dijkstra will choose P2 since the length of  $P2 = \max(5/12, 5/12) = 5/12 < \text{length of } P1 = \max(10/12, 1/12) = 10/12$ . However, the shortest path from the source to the destination is through P1, since  $\max(11/12, 11/12) < \max(6/12, 15/12)$ .

### 3.3.2 Delay Constrained Least Cost Routing (DCLC)

The delay-constrained least-cost routing problem is a type of path-constrained path-optimization problem and can be defined as follows: Given a network graph  $G = (V, E)$ , with  $V$  nodes and  $E$  edges, non-negative cost  $C(e)$  and delay  $D(e)$  for each edge  $e \in E$ , a source  $s$  and a destination  $d$ , and a positive delay constraint  $\Delta_d$ , find a path  $P$  whose cost is minimum among the paths whose delay is less than the delay constraint,  $\Delta_d$ .

Among the proposed heuristics, we will look at the Delay-Cost-Constrained Routing with Search Space Reduction (SSR+DCCR) algorithm [6] and Delay Constrained Unicast Routing (DCUR) algorithm [14]. Guo and Matta [6] suggest converting the problem into a delay-cost-constraint problem and using a nonlinear path length definition (as explained in Section 3.3.1) which gives priority to low cost paths. By doing this, a path-constrained path-optimization problem is converted into a multiple constraints routing problem. The algorithm also uses a linear path length definition in the pre-processing step to be able to find a tight enough bound for the (introduced) cost constraint. This approach effectively reduces the search space and increases the chance of finding the optimal path. The DCUR algorithm [14] is a distributed heuristic solution, where each node keeps cost and delay vectors. Cost/Delay vectors keep for each destination the next-hop on the shortest path. Each node independently decides whether to choose the least delay next-hop or the least cost next-hop. The least cost path is preferred if the sum of the delay along the path that has been chosen so far, the delay to the next-hop node along the least cost path, and the delay of the least delay path from the next-hop to the destination satisfies the delay constraint. The algorithm is distributed and uses distance vector, which makes it more scalable. Although the run time complexity is very low, the communication complexity is increased. Communication complexity can be reduced at the cost of increased space complexity if periodically advertised cost and delay vectors from neighbors are saved. By being a QoS routing algorithm, DCUR still needs to keep per-flow state.

Per-flow state limits the ability to scale, and all the QoS al-

gorithms we have seen so far assume that it is essential for QoS routing to provide performance guarantees. Mieghem *et al.* [12] question this way of thinking and try to see if it is possible to achieve sufficient level of QoS guarantees using hop-by-hop destination-based only routing (connectionless). They conclude that it is possible if each packet carries either the constraints that must be satisfied for the rest of the path or the path traversed so far and the original constraints. Even with this modification, the algorithm still requires link state routing and carrying information in each packet increases per-packet cost.

### 3.4 Improving Long Term Utilization

Long term utilization of the network can be improved by choosing paths that use network resources in an efficient way. Improved long term utilization manifests itself in the form of increased revenue and decreased blocking probability. In the context of QoS routing, long term utilization can be improved by using additional constraints to limit resource consumption, balance or pack load. Ma and Steenkiste [10] suggest preferring shortest (min-hop) paths to limit resource consumption and show that under heavy load doing so improves performance. Load balancing improves performance under light load and can be achieved by preferring least loaded paths (widest path routing). Matta and Bestavros [11] show that load balancing increases the blocking probability of high bandwidth requests because of bandwidth fragmentation. Load packing is another way to improve long term utilization, where the most utilized paths are preferred to approximate perfect fit, which is NP-hard. This way, bandwidth fragmentation is minimized and fairness for large bandwidth requests is improved. However, load packing is extremely sensitive to link state inaccuracies since the paths with very tight resource availability are targeted.

## 4. Traffic Aware Routing

The algorithms that fall in this category have some knowledge of traffic in the form of the location of ingress-egress pairs and/or the traffic demands between them (i.e. the traffic matrix). The algorithms may or may not have information about the QoS requirements of individual requests. The goal is to use this extra information to optimize/maximize network usage or service guarantees. For example, knowing the demands between each source-destination pair allows the pre-computation of a set of optimal routes.

### 4.1 How performance can be improved if location of ingress-egress pairs are known?

Knowledge of ingress-egress pairs can be used along with QoS routing (Section 3) to improve utilization and long term performance. For bandwidth constrained on-demand

routing where there is no knowledge of future requests, and splitting of individual requests is not allowed, Kar *et al.* [7] suggest picking feasible paths that interfere least with future requests. Figure 4 shows an example. When S3 wants

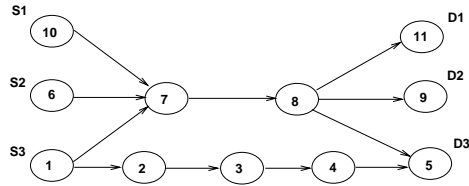


Figure 4. From S3 to D3, 1-2-3-4-5 is the minimum interference path.

to communicate with D3, either  $P_1(1,7,8,5)$  or  $P_2(1,2,3,4,5)$  can be picked. However, if S3 knows of other ingress-egress pairs (S2,D2) and (S1,D1), then it can avoid using path  $P_1$  so that the link (7,8) will stay available for their future requests. Before we explain how minimum interference paths are computed, we should define maximum flow. The maximum flow is an upper bound on the total amount of bandwidth that can be routed between an ingress-egress pair. A linear programming formulation cannot be used to maximize the sum of maximum flows between other ingress-egress pairs because of splittability restrictions. Therefore, the problem is NP-hard. Kar *et al.* [7] propose a heuristic which is called **Minimum Interference Routing Algorithm (MIRA)**. The idea is to assign link weights based on their criticality. A link is critical if it belongs to a min-cut for an ingress-egress pair. In other words, routing over a critical link would decrease the maxflow value of some ingress-egress pair and we should defer loading such kind of links as much as possible. After assigning link weights, a minimum interference route can be found using Dijkstra's shortest path algorithm.

### 4.2 How performance can be improved if traffic matrix is known?

One way to use the traffic matrix is to find the **optimal solution**. This can be done by solving a linear programming formulation to optimize a metric like minimizing the maximum link utilization. This approach allows unrestricted splitting of individual requests and the resulting optimal routes can only be realized by setting up logical connections. Therefore, this approach is costly, since it requires more than per-flow state in case of splits. This limits the scalability of this approach.

The traffic matrix can be used to calculate **OSPF weights** in such a way that shortest path routing would prefer under-utilized links. The main advantage of this approach is its simplicity, eliminating the need to keep per-flow state in the form of logical paths. Since finding an optimal weight setting for OSPF-ECMP is NP-hard, heuristics are suggested. Fortz and Thorup [4] propose a local search

approach. The idea is to search for a weight setting that will result in such a load distribution whose total cost is minimum, where the cost function is defined to be a piece-wise linear increasing function of load. Another heuristic proposed in [9] suggests setting link weights as an exponential function of link load.

Both [4] and [9] prove that weight setting for OSPF-ECMP cannot replace per-flow routing as a traffic engineering tool. Specifically, it has been shown that OSPF-ECMP is  $\Omega(N)$  worse than per-flow routing with respect to maximum throughput and maximum utilization that can be achieved, where  $N$  is number of flows. However, practical aspects such as whether the worst case gap is observed in practice for realistic topologies or traffic patterns have not been studied.

Another way to use the traffic matrix is to maximize the probability of admitting future requests through **load profiling** [11]. Load profiling is a probabilistic heuristic suggested as a more robust alternative for most loaded routing (Section 3.4), which is extremely sensitive to link state inaccuracies. The idea is to keep the input load profile and the bandwidth availability profile across the candidate paths as close as possible so that each incoming request with diverse bandwidth requirements will be able to find a feasible path. The approach effectively eliminates the high blocking probability of high bandwidth requests.

The traffic matrix can be used along with ingress-egress pair information in the context of QoS routing to increase utilization and long term performance. An example of this is **Profile-based Routing (PBR)** [18]. On-demand routing is assumed where no knowledge of future requests is available and splitting of flows is not allowed. A traffic profile is defined to be the aggregate expected traffic between an ingress-egress pair for a particular class. The algorithm has two phases: off-line (preprocessing) and on-line. During the off-line phase, by using a linear programming formulation, the optimal distribution of traffic profiles is computed. In the second (on-line) phase, the result of the off-line phase is used as virtual capacities to route individual requests on demand. The off-line phase serves as an admission control mechanism: A request maybe rejected even if there is a feasible path had actual capacities been used.

Traffic demands are based on long-term averages such as daily demands. Therefore, the solutions based on the usage of traffic matrix require re-optimization from time to time. In the meantime, the algorithm is static. The traffic matrix information is generally used off-line by centralized algorithms, which are not designed to handle traffic fluctuations in real-time.

While we are using the traffic matrix to optimize long term performance, what happens to short term performance? Sridharan *et al.* [17] show that short-term performance maybe sub-optimal. Traffic variations at shorter time scales may cause temporary overloads and computing a new set of optimal routes maybe impractical at this frequency. Increasing the flow granularity helps to reduce short term traffic

variability. However, coarser flow granularity doesn't allow optimal traffic distribution, since it forces traffic to remain aggregated, which can result in poorer long term performance.

## 5. Does increased complexity mean better performance?

Yilmaz and Matta [19] compare the performance of Widest Shortest Path (WSP), Minimum Interference Routing (MIRA), Profile-based routing (PBR), and per-packet dynamic routing for bandwidth acceptance ratio, and maximum link utilization. The goal is to observe the effects of increased complexity on performance. Each of these algorithms represents a different class in our taxonomy. Per-packet dynamic routing is only aware of available resources, therefore it is a best-effort routing algorithm. WSP is a QoS routing algorithm, for guaranteed bandwidth, so that in addition to resource availability, it is also aware of the bandwidth requirements of individual requests. Both MIRA and PBR are traffic aware routing algorithms where MIRA only uses ingress-egress pair information while PBR uses both ingress-egress pair information and the traffic demand matrix. Both MIRA and PBR are also QoS routing algorithms, for guaranteed bandwidth, where the bandwidth requirement of individual flows are known. Per-packet dynamic routing is simple, distributed, stateless, and scalable. However, it maybe hard to deploy in practice because of potential routing oscillations. Also, it cannot provide any guarantees on service quality. WSP, MIRA and PBR keep per-flow state, use source and link state routing. WSP only requires Dijkstra's shortest path algorithm, therefore it is computationally not expensive. MIRA is computationally expensive, and has to maintain the location of ingress-egress pairs. PBR is the most expensive one in terms of the information maintained and stored. It also requires re-optimization of the traffic profile distribution from time to time, which requires re-routing of existing flows, and keeping track of changes in traffic profiles. In conclusion, the simplest and most scalable solution is per-packet dynamic routing whereas the most expensive and least scalable of all is PBR. PBR is the only algorithm that uses all the information available. Properties and complexities of the algorithms are shown in Table 1.

The questions to be asked are: How close are we getting to the optimal per-packet dynamic routing? Does increased complexity and usage of more information in terms of QoS requirements, traffic matrix etc. help improve performance? Is the performance gain good enough to sacrifice scalability?

These questions are answered through simulations. Per-packet dynamic routing is simulated by solving the multicommodity flow problem at each flow arrival/departure where each flow is a commodity. Excess edges are added to always have a feasible solution. A cost of infinity is assigned to excess edges while a cost of 1 is assigned to orig-

	Widest-Shortest Path [5]	Minimum Interference Routing [7]	Profile-based Routing [18]	Dynamic per-packet routing
Solves	Guaranteed bandwidth	Guaranteed bandwidth	Guaranteed bandwidth	Best Effort
Routing Strategy	Source	Source	Source	Hop-by-hop
Type	Link State	Link State	Link State	Link State or Distance Vector
Time Complexity	$O(E \log V)$ [3]	$O(nVE^2)$ n is number of ingress-egress pairs	$O(V+E)$ for on-line phase +offline phase	$O(1)$
Communication Complexity	$O(k)$ to distribute link states k is number of neighbors	$O(k)$ to distribute link states k is number of neighbors	$O(k)$ to distribute link states k is number of neighbors	$O(k)$ to distribute link states if Link State k is number of neighbors $O(V)$ if Distance Vector
Space Complexity	$O(E)$ for network state	$O(E)$ for network state $O(V^2)$ for ingress-egress pair matrix	$O(E)$ for network state $O(V^2)$ for ingress-egress pair matrix $O(V^2)$ for traffic profile matrix	$O(E)$ for network state if Link State $O(kV)$ for path state if Distance Vector k is number of neighbors
Maintained State	per-flow	per-flow	per-class and per-flow	none
Extra Information Used	none	ingress-egress pair matrix	ingress-egress pair matrix traffic profile matrix	none

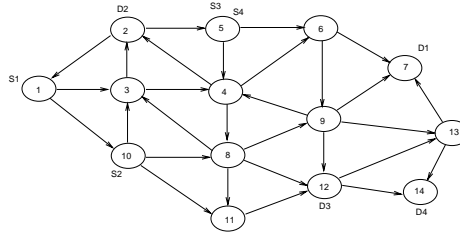
**Table 1. Algorithms and their properties.**

inal edges. To maximize the traffic routed along the original edges, the following linear programming formulation is used:

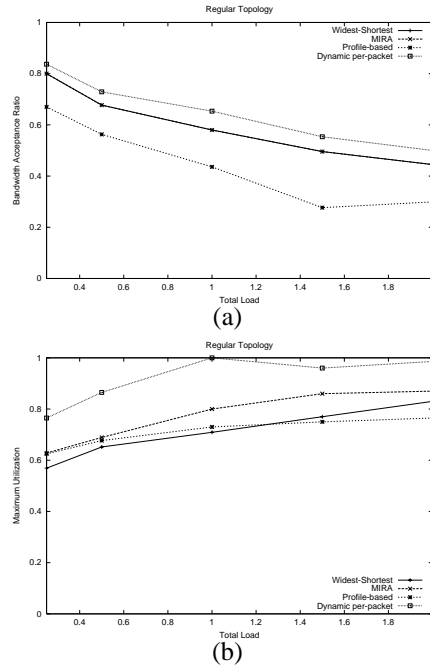
$$\text{minimize } \sum (cost(e) \sum x_i(e))$$

subject to capacity and flow conservation (except at ingress and egress nodes) constraints. The output of this linear programming formulation is the values of variables  $x_i(e)$  which denote the amount of commodity  $i$  that is routed through edge  $e$ . This version of the multicommodity flow problem optimizes bandwidth acceptance ratio while preferring shorter paths at the expense of load imbalances. The main purpose of widest-shortest path routing algorithm is to balance load. Therefore, maximum link utilization should be lowest with WSP. Figure 5 shows a topology on which the simulation is done. The number of ingress-egress pairs are 3, and all the link capacities are 150. The *bandwidth acceptance ratio* and *maximum utilization* plots are shown in Figure 6. MIRA and WSP have the same *bandwidth acceptance ratio*, which is very close to that of per-packet dynamic routing. PBR has the worse performance. The main problem with PBR is imposing admission control at the beginning (off-line phase) and saving resources for flows that do not arrive for some time. This causes statistical multiplexing loss. In terms of *maximum utilization*, dynamic per-packet routing has the highest value, because of packing shortest paths first. WSP achieves good load balancing, better than MIRA, while providing the same *bandwidth acceptance ratio*. This is because the main concern of WSP is to balance load, while with MIRA, the main goal is to maximize open capacity of other ingress-egress pairs. PBR also has very low maximum utilization, merely because it lets links go under-utilized due to its admission control. This admission control is based on virtual capacities computed in the off-line phase.

Overall, it is observed that for different topologies, PBR cannot achieve the performance of MIRA and WSP, because of loss of statistical multiplexing. WSP performs as well as MIRA, and balances load better. Furthermore, while the simplest of the per-flow algorithms we consider, the performance of WSP is close to dynamic per-packet routing, without the potential instabilities and lack of service guarantees of dynamic per-packet routing.



**Figure 5. Regular Topology**



**Figure 6. (a) Bandwidth Acceptance Ratio for Regular Topology, (b) Maximum Utilization for Regular Topology.**

## 6. Future Directions

Future unicast routing solutions are expected to be hybrid schemes where cost is manageable and performance is satisfactory. The solutions that are at the extreme ends of the spectrum either do not provide good enough performance but are scalable or provide good performance but do

not scale hence of no practical value. Scalability problems of per-flow routing can be solved by aggregation. Classifying flows at the edges based on their characteristics and handling traffic classes inside the core is an efficient way to balance scalability and performance. While DiffServ architectures efficiently use this method at the link scheduling level to find a good operating point between IntServ architectures and best effort, routing algorithms have to be extended such as each class of traffic can be routed depending on its characteristics.

Another challenging aspect of unicast routing is finding good heuristics for multiple constraints routing and path-constrained path-optimization problems. The proposed heuristics should be able to find feasible solutions without requiring expensive computations.

Although stability is challenging for both dynamic and policy-based routing, extending BGP (inter-domain) routing to support QoS requires careful examination of cost-performance tradeoffs.

Since it is impractical to have accurate link state information, routing algorithms are being developed to work well under stale link state information either by estimating actual link state values or by being robust to such kind of conditions.

## References

- [1] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi. "Quality of Service Routing: A Performance Perspective". In *Proceedings of ACM SIGCOMM*, Vancouver, British Columbia, Canada, August 1998.
- [2] S. Chen and K. Nahrstedt. "An Overview of Quality of Service Routing for Next Generation High-Speed Networks: Problems and Solutions". In *IEEE Networks*, 12(6):64-79, November/December 1998.
- [3] T. Cormen, C. Leiserson, and R. Rivest. "Introduction to Algorithms". MIT Press, Cambridge, MA, 1990.
- [4] B. Fortz and M. Thorup. "Internet Traffic Engineering by Optimizing OSPF Weights". In *Proceedings of IEEE INFOCOM*, Tel-Aviv, Israel, March 2000.
- [5] R. Guerin, A. Orda, and D. Williams. "QoS Routing Mechanisms and OSPF Extensions". In *Globecom*, Phoenix, AZ, November 1997.
- [6] L. Guo and I. Matta. "Search Space Reduction in QoS Routing". In *Proceedings of the 19th International Conference on Distributed Computing Systems*, Austin, Texas, June 1999. Extended version in *Computer Networks Journal*, 2002.
- [7] K. Kar, M. Kodialam, and T. Lakshman. "Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications". In *Proceedings of IEEE INFOCOM*, Tel-Aviv, Israel, March 2000.
- [8] A. Khanna and J. Zinky. "The Revised ARPANET Routing Metric". In *Proceedings of ACM SIGCOMM*, Austin, Texas, September 1989.
- [9] D. Lorenz, A. Orda, D. Raz, and Y. Shavitt. "How good can IP routing be?". In *DIMACS Technical Report 2001-17*, May 2001.
- [10] Q. Ma and P. Steenkiste. "On Path Selection for Traffic with Bandwidth Guarantees". In *Proceedings of ICNP: IEEE International Conference on Network Protocols*, Atlanta, Georgia, October 1997.
- [11] I. Matta and A. Bestavros. "A Load Profiling Approach to Routing Guaranteed Bandwidth Flows". In *Proceedings of IEEE INFOCOM*, March 1998. Extended version in *European Transactions on Telecommunications - Special Issue on Architectures, Protocols and Quality of Service for the Internet of the Future*, March-April 1999.
- [12] P. Mieghem, H. Neve, and F. Kuipers. "Hop-by-hop QoS Routing". In *Computer Communications* 37 (2001) 407-423, 2001.
- [13] V. Paxson. "End-to-End Routing Behavior in the Internet". In *Proceedings of ACM SIGCOMM*, Stanford, CA, August 1996.
- [14] H. Salama, D. Reeves, and Y. Viniotis. "A Distributed Algorithm for Delay-Constrained Unicast Routing". In *Proceedings of IEEE INFOCOM*, Kobe, Japan, 1997.
- [15] S. Savage, A. Collins, and E. Hoffman. "The End-to-End Effects of Internet Path Selection". In *Proceedings of ACM SIGCOMM*, Cambridge, MA, September 1999.
- [16] A. Shaikh, J. Rexford, and K. Shin. "Load-sensitive Routing of Long-lived IP Flows". In *Proceedings of ACM SIGCOMM*, Boston, MA, September 1999.
- [17] A. Sridharan, S. Bhattacharyya, R. Guerin, J. Jethava, and N. Taft. "On Impact of Aggregation on the Performance of Traffic Aware Routing". In *ITC'17*, September 2001.
- [18] S. Suri, M. Waldvogel, and P. Warkhede. "Profile-based Routing: A New Framework for MPLS Traffic Engineering". In *Quality of Future Internet Services, Lecture Notes in Computer Science 2156*, September 2001.
- [19] S. Yilmaz and I. Matta. "On the Scalability-Performance Tradeoffs in MPLS and IP Routing". In *Proceedings of SPIE: Scalability and Traffic Control in IP Networks*, July 2002.