Traffic Engineering Techniques and Algorithms for the Internet

Debashis Basak[1], Hema Tahilramani Kaur, Shivkumar Kalyanaraman[2]
Email: *D.Basak@accelight.com, {hema,shivkuma}@networks.ecse.rpi.edu*

Abstract

Traffic engineering broadly relates to optimization of the operational performance of a network. This survey discusses techniques like multi-path routing, traffic splitting, constraint-based routing, path-protection etc. that are used for traffic engineering in contemporary Internet Service Provider (ISP) networks. These techniques can be classified under two broad classes, connectionless and connection-oriented, that dominate the current debate on next-generation routing and traffic engineering in IP networks. The connectionless approach evolves current distance-vector and link-state algorithms, or influences routing metrics. The connection-oriented approach uses signaling and is being used by techniques like Multi Protocol Label Switching (MPLS). Connection-oriented techniques offer a convenient way to monitor, allocate, reroute, and protect resources for a given traffic on an explicit and flexible basis. This survey will examine the core problems, discuss solutions in both connectionless and signaled approach, and point to topics for research and advanced development.

[1] Debashis Basak is with Accelight Networks

[2] Hema Tahilramani Kaur and Shivkumar Kalyanaraman are with Rensselaer Polytechnic Institute, Dept of ECSE.

1 Introduction

The unprecedented growth of the Internet has lead to a growing challenge among the ISPs to provide a good quality of service, achieve operational efficiencies and differentiate their service offerings. ISPs are rapidly deploying more network infrastructure and resources to handle the emerging applications and growing number of users. Enhancing the performance of an operational network, *at both the traffic and the resource levels*, are major objectives of traffic engineering [3]. Traffic engineering (TE) is defined as *... that aspect of Internet network engineering dealing with the issue of performance evaluation and performance optimization of operational IP networks...* [3]. The goal of performance optimization of operational IP networks is accomplished by routing traffic in a way to utilize network resources efficiently and reliably. Traffic engineering has been used to imply a range of objectives, including load-balancing, constraint-based routing, multi-path routing, fast re-routing, protection switching etc.

1.1 Network Planning, Network Engineering, Traffic engineering

A number of terms are used in the literature to characterize the network operational functions. *Network planning* is a long-term process used to build a physical network for long-term traffic growth. *Network engineering* is another process that uses dynamic reconfiguration of links according to the status of the networks, a property that is supported by dynamically configurable circuit-switched networks. *Traffic engineering* is a shorter-term process used to optimize network resource utilization for traffic demands. In other words, while traffic engineering aims to map traffic to available capacity (on a relatively static topology), network engineering aims to establish capacity where the traffic needs it.

1.2 Traffic Engineering Functions

Awduche et al [2] note that a distinctive function performed by Internet traffic engineering is the control and optimization of the routing function, to steer traffic through the network in the most effective way. Traffic engineering also attempts to optimize the characteristics of the network that are visible to users (a.k.a "*emergent properties*") while taking economic considerations into account. The control function of TE can take two forms: proactive and reactive. A proactive TE control system takes preventive action to obviate predicted unfavorable future network states. A reactive TE control system responds correctively and perhaps adaptively to events that have already transpired in the network. Finally, measurement is a critical function of traffic engineering for operational, accounting and billing reasons.

The optimization function of traffic engineering can be achieved through *capacity management* and *traffic management*. Capacity management includes capacity planning, routing control, and resource management. Network resources of particular interest include link bandwidth, buffer space, and computational resources. Likewise, traffic management includes (a) nodal traffic control functions such as traffic conditioning, queue management, scheduling, and (b) other functions that regulate traffic flow through the network or that arbitrate access to network resources between different packets or between different traffic streams. Constraint-based routing is a generalization of QoS routing that take specified traffic attributes, network constraints, and policy constraints into account when making routing decisions. Constraint-based routing is applicable to traffic aggregates as well as flows.

The control function of Internet traffic engineering responds at multiple levels of temporal resolution to network events. Certain aspects of capacity management, such as capacity planning, respond at very coarse temporal levels, ranging from days to possibly years. The introduction of automatically switched optical transport networks (e.g. based on the Multi-protocol Lambda Switching concepts) could significantly reduce the lifecycle for capacity planning by expediting provisioning of optical bandwidth. Routing control and packet level processing functions operate at finer levels of temporal resolution.

The measurement function is crucial to TE because the operational state of a network can be conclusively determined only through measurement. Measurement is also critical to the optimization function because it provides feedback data that is used by traffic engineering control subsystems. This data is used to adaptively optimize network performance in response to events and stimuli originating within and outside the network. Measurement is also needed to determine the quality of network services and to evaluate the effectiveness of traffic engineering policies as perceived by users, i.e., emergent properties of the network.

1.3 Hop-by-hop Vs Signaled Routing Models[3]

Next-generation routing capabilities are one of the primary mechanisms used to achieve traffic engineering objectives. Two models dominate the current debate on next-generation routing: *hop-by-hop* and the *signaled* models.

The *hop-by-hop* model (a.k.a connectionless model) relies on extending existing Link-State (LS), Distance-Vector (DV) or Path-Vector (PV) algorithms. Optimally setting link weights and using various multi-path algorithms are some examples of the connectionless approach to TE. The advantage of this approach is its simplicity that leads to scalability and ease of inter-networking. The disadvantage of this approach is that the TE capabilities are achieved as a side effect of setting weights, i.e., *indirectly*. Also the single-shortest-path nature of most deployed protocols limits the range of TE capabilities achievable.

The *signaled* model (a.k.a connection-oriented model) first signals the establishment of the entire path and reserves resources before sending packets. The two-phase, explicit-path selection and setup nature of this model allows a range of traffic engineering capabilities to be *directly* achieved. However, the cons of this approach include its need for a signaling protocol and VC-setup prior to transmission that complicates its mapping to IP routing protocols like BGP and OSPF. Hence, this model has been primarily used inside large service provider ASs and not between ASs. This model has been implemented in technologies like MPLS [50], ATM and frame-relay.

Routing has two types of operations: *data-plane* and *control-plane* operations. The data-plane operations are those that are performed on every packet (eg: address matching, forwarding etc),

[3] A.k.a connectionless vs connection-oriented routing models

whereas the control-plane sets up information (eg: route-table setup, signaling) to facilitate data-plane operations.

In the *hop-by-hop* (or connectionless) model, local knowledge is distributed to immediate neighbors, and ultimately reaches all nodes. Every node infers routes based upon this information. A consistency criterion ensures that independent decisions made by nodes lead to valid, loop-free routes. The data-plane forwarding algorithm in this model is related to the control-plane algorithm because both use the same *global* identifiers (e.g. IP addresses, prefixes, link metrics, AS numbers). This relationship or *coupling* has, in the past, required changes in the forwarding algorithm whenever the control-plane algorithm was significantly changed (e.g. subnet masking, CIDR) [12]. However, routing protocols based on the hop-by-hop model dominate the control-plane of the Internet (e.g. RIP, EIGRP, OSPF [44], IS-IS, BGP [55]) for three important reasons; they support connectionless forwarding, they can be inter-networked easily, and they scale reasonably well. Traffic engineering efforts using this model have mostly focused on optimizing the performance via parametric or indirect methods. Protocol extensions for traffic engineering in this model have not been deployed because they require major upgrades due to the coupling of routing data- and control-planes. We discuss connectionless TE work in greater detail in Section 3.

In the *signaled* (or connection-oriented) model, local information may be sent to all nodes through an approach similar to hop-by-hop algorithms. However, it is the source node or some central entity that computes the desired paths and decides what traffic is mapped to those paths. The intermediate nodes (or switches) then set up local path identifiers (called *labels* in MPLS) for the paths. The signaling protocol allows autonomy in the choice of labels at switches, but ensures the consistency between label assignments at adjacent switches in the path. This leads to a label-switching forwarding algorithm where labels are switched at every hop. The forwarding algorithm in the signaled model (ATM, MPLS) is de-coupled from the control algorithms. This is because the forwarding algorithm uses *local* identifiers (labels), whereas the control algorithms use *global* identifiers (addresses). The signaling protocol maps and ensures consistency between local and global identifiers. This *de-coupling* between forwarding and control-planes allows the introduction of new TE capabilities by modifying the control plane alone. However, signaled approaches have historically been hard to inter-network (e.g. IP over ATM [41], Non-Broadcast Multiple Access (NBMA) routing [44] or multi-domain signaled TE), and hence have been limited to intra-domain or intra-area deployments (e.g. MPLS, ATM). In fact, most of the work in the area of TE has focused on a *single, flat routing domain*. However, due to connection-oriented nature of the signaled model, it is possible to improve reliability of network operations and provide more features such as QoS assurances in terms of bandwidth, packet loss rate etc. MPLS is a solution based on the signaled approach, which is being deployed very rapidly. Therefore, in this paper we focus on recent developments in MPLS for addressing various TE objectives such as QoS, constraint-based routing, path protection etc.

We conjecture that the key reasons for the lag in broad adoption of connectionless TE capabilities include the need for complete network upgrades, lack of source-based or explicit operator control over TE decisions, lack of a common reference framework that allows long-term evolution of TE capabilities. This is because today connectionless protocols only allow an indirect (or backdoor) approach to TE. Connection-less TE enjoys some deployment because it allows the leverage of already-installed routing protocols like OSPF, IS-IS etc. On the other hand, a signaled framework (like MPLS) provides a substantial set of features that can be directly leveraged to form the basis of sophisticated TE implementations in ISP networks.

2 Routing Algorithms, Protocols, Frameworks: Overview

Routing is the magic enabling connectivity. It is the control-plane function, which sets up the local forwarding tables at the intermediate nodes, such that a concatenation of local forwarding decisions leads to global connectivity. The global connectivity is also "efficient" in the sense that loops are avoided in the steady state.

Internet routing is scalable because it is hierarchical. There are two categories of routing in the Internet: inter-domain routing and intra-domain routing. *Inter*-domain routing is performed between autonomous systems (AS's). An autonomous system defines the locus of single administrative control and is internally connected, i.e., employs appropriate routing so that two internal nodes need not use an external route to reach each other. The internal connectivity in an AS is achieved through *intra*-domain routing protocols. Once the nodes and links of a network are defined and the boundary of the routing architecture is defined, then the routing protocol is responsible for capturing and condensing the appropriate global state into local state (i.e. the forwarding table). Two issues in routing are *completeness* and *consistency*.

In the steady state, the routing information at nodes must be *consistent*, i.e., a series of independent local forwarding decisions must lead to connectivity between any (source, destination) pair in the network. If this condition is not true, then the routing algorithm is said to not have "*converged*" to steady state, i.e., it is in a transient state. In certain routing protocols, convergence may take a long time. In general a part of the routing information may be consistent while the rest may be inconsistent. If packets are forwarded during the period of convergence, they may end up in loops or arbitrarily traverse the network without reaching the destination. This is why the TTL field in the IP header is used. In general, a faster convergence algorithm is preferred, and is considered more stable; but this may come at the expense of complexity. Longer convergence times also limit the scalability of the algorithm, because with more nodes, there are more routes, and each could have convergence issues independently.

Completeness means that every node has sufficient information to be able to compute all paths in the entire network locally. In general, with more complete information, routing algorithms tend to converge faster, because the chances of inconsistency reduce. But this means that more distributed state must be collected at each node and processed. The demand for more completeness also limits the scalability of the algorithm. Since *both* consistency and completeness pose scalability problems, large networks have to be structured hierarchically (eg: as areas in OSPF) where each area operates independently and views the other areas as a single border node.

2.1 Hop-by-hop Routing Protocols

The two main types of hop-by-hop routing are link-state and distance vector. *Distance vector* protocols maintain information on a *per-node* basis (i.e. a vector of elements), where each element of the vector represents a distance or a path to that node. *Link state* protocols maintain information on a *per-link* basis where each element represents a weight or a set of attributes of a link. If a graph is considered as a set of nodes and links, it is easy to see that the link-state approach has complete information (information about links also implicitly indicates the nodes which are the end-points of the links) whereas the distance vector approach has incomplete information. The basic algorithms of the distance vector (Bellman-Ford) and the link-state (Dijkstra) attempt to find the shortest paths in a graph, in a fully distributed manner, assuming that distance vector or link-state information can only be exchanged between immediate neighbors. Both algorithms rely on a simple recursive consistency criterion.

2.1.1 Distance Vector (DV) Algorithm

Assume that the *shortest* distance path from node i to node j has (shortest) distance D(i,j), and it passes through neighbor k to which the cost from i is c(i,k), then we have the equation (See Figure 1):

$$\mathbf{D(i,j) = c(i,k) + D(k,j)} \qquad \mathbf{(1)}$$

In other words, equation (1) is a special case of a more general consistency criterion that "…*the subset of a shortest path is also the shortest path between the two intermediate nodes...*"

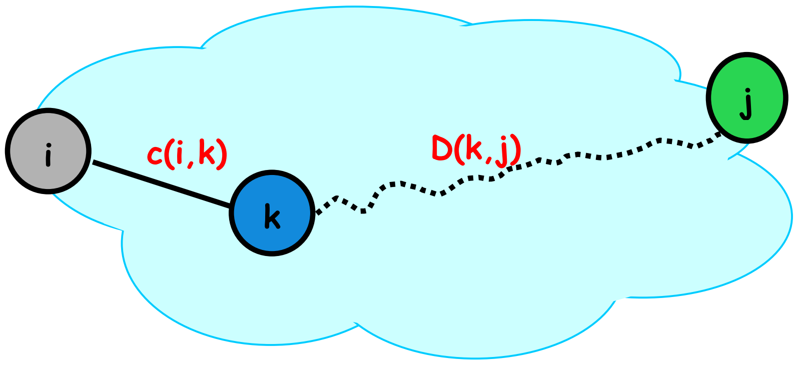

Figure 1: Physical Meaning of the Consistency Criterion in DV algorithms

The *distance vector (Bellman-Ford) algorithm* evaluates the above recursion iteratively by starting with initial distance values:

D(i,i) = 0 ;
D(i,k) = c(i,k) if k is a neighbor (i.e. k is one-hop away); and
D(i,k) = INFINITY for all other non-neighbors k.

Observe that the set of values D(i,*) is a *distance vector at node i*. The algorithm also maintains a nexthop value for every destination j, initialized as:

> **next-hop(i) = i;**
> **next-hop(k) = k** if k is a neighbor, and
> **next-hop(k) = UNKNOWN** if k is a non-neighbor.

Note that the next-hop values at the end of every iteration go into the forwarding table used at node i.

In every iteration each node i exchanges its distance vectors D(i,*) with its immediate neighbors. Now each node i has the values used in equation (1), i.e. D(i,j) for any destination and D(k,j) and c(i,k) for each of its neighbors k. Now if c(i,k) + D(k,j) is smaller than the current value of D(i,j), then D(i,j) is replaced with c(i,k) + D(k,j), as per equation (1). The next-hop value for destination j is set now to k. Thus after *m* iterations, each node knows the shortest path possible to any other node which takes *m* hops or less. Therefore the algorithm converges in O(d) iterations where d is the maximum diameter of the network. Observe that each iteration requires information exchange between neighbors. At the end of each iteration, the next-hop values for every destination j are output into the forwarding table used by IP.

2.1.2 Link State (LS) Algorithm

The *link state (a.k.a. Dijkstra) algorithm* pivots around the final link cost c(k,j) and the destinations j, rather than the distance D(i,j) and the source i in the distance-vector approach. In particular, the consistency criterion in link-state algorithm (see Figure 2) is:

$$\mathbf{D(i,\ j)\ =\ D(i,k) + c(k,j)} \qquad \mathbf{(2)}$$

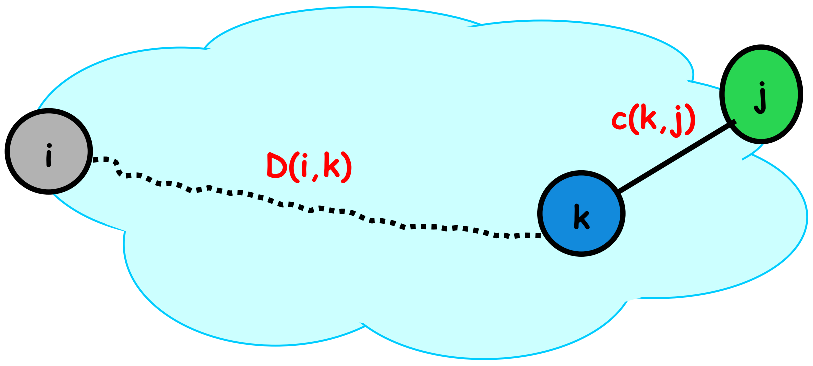

Figure 2: Physical Meaning of the Consistency Criterion in LS algorithms

The LS algorithm follows a greedy iterative approach to evaluating (2), but it collects all the link states in the graph *before* running the Dijkstra algorithm *locally*. The Dijkstra algorithm at node i maintains two sets: set N that contains nodes to which the shortest paths have been found so far, and set M that contains all other nodes. Initially, the set N contains node i only, and the next hop (i) = i. For all other nodes k a value D(i,k) is maintained which indicates the current value of the path cost (distance) from i to k. Also a value p(k) indicates what is the predecessor node to k on the shortest known path from i (i.e. p(k) is a neighbor of k).

Initially,

> **D(i,i) = 0   and   p(i) = i;**
> **D(i,k) = c(i,k)   and   p(k) = i** if k is a neighbor of i
> **D(i,k) = INFINITY   and   p(k) = UNKNOWN** if k is *not* a neighbor of i
> Set N contains node i only, and the next hop (i) = i.
> Set M contains all other nodes j.

In each iteration, a new node j is moved from set M into the set N. Such a node j has the minimum distance among all current nodes in M, i.e. D(i,j) = min $_{\{l \varepsilon M\}}$ D(i,l). If multiple nodes have the same minimum distance, any one of them is chosen as j. Node j is moved from set M to set N, and the next-hop(j) is set to the neighbor of i on the shortest path to j. Now, in addition, the distance values of any neighbor k of j in set M is reset as:

> **If D(i,k) < c(j,k) + D(i,j), then D(i,k) = c(j,k) + D(i,j), and p(k) = j.**

This operation called "*relaxing*" the edges of j is essentially the application of equation (1). This defines the end of the iteration. Observe that at the end of iteration *p* the algorithm has effectively explored paths, which are *p* hops or smaller from node i. At the end of the algorithm, the set N contains all the nodes, and knows all the next-hop(j) values which are entered into the IP forwarding table. The set M is empty upon termination. The algorithm requires *n* iterations where *n* is the number of nodes in the graph. But since the Dijkstra algorithm is a *local* computation, they are performed much quicker than in the distance vector approach. The complexity in the link-state approach is largely due to the need to wait to get all the link states c(j,k) from the entire network.

2.1.3 Protocols: RIP, OSPF, BGP-4

The protocols corresponding to the distance-vector and link-state approaches for *intra*-domain routing are RIP, EIGRP (DV) and OSPF, IS-IS (LS) respectively. In both these algorithm classes if a link or node goes down, the link costs or distance values have to be updated. Hence information needs to be distributed and the algorithms need to be rerun. RIP is used for fairly small networks mainly due to a convergence problem called "*count-to-infinity.*" The advantage of RIP is simplicity. OSPF is a more complex standard that allows wider set of link weights, efficient mappings to a variety of subnets, hierarchical routing and is in general more stable than RIP. Therefore it is used in larger networks (esp enterprise and ISP internal networks). Another popular link-state protocol commonly used in ISP networks is IS-IS, which came from the ISO/OSI world, but was adapted to IP networks.

BGP-4 is the *inter*-domain protocol of the Internet. It uses a vectoring approach, but uses full AS-paths instead of distances used in RIP. BGP-4 is designed for policy based routing between autonomous systems, and therefore it does not use the Bellman-Ford algorithm. BGP speakers announce routes to certain destination prefixes expecting to receive traffic that they then forward along. When a BGP speaker receives updates from its neighbors that advertise paths to destination prefixes, it assumes that the neighbor is actively using these paths to reach that destination prefix. These route advertisements also carry a list of *attributes* for each prefix. BGP then uses a list of tiebreaker rules (like a tennis tournament) to determine which of the multiple paths available to a destination prefix is chosen to populate the forwarding table. The tiebreaker rules are applied to attributes of each potential path to destination prefixes learnt from neighbors. The AS-path length is one of the highest priority items in the tiebreaker process. Other attributes include local preference, multi-exit-discriminator (aka LOCAL_PREF, MED: used for redundancy/load balancing), ORIGIN (indicates how the route was injected into BGP), NEXT-HOP (indicates the BGP-level next hop for the route) etc.

2.2 Signaled Routing Frameworks: ATM and PNNI

The asynchronous transfer mode (ATM) technology was a culmination of several years of evolution in leased data-networks that grew out from the telephony world. ATM was developed as a convergence technology where voice and data could be supported on a single integrated network. ATM lost out in the LAN space because of the emergence of Fast Ethernet in the early 90s and Gigabit Ethernet in the mid-to-late 90s. ATM never became the basis for end-to-end transport, (i.e. native ATM applications are rare) because the TCP/IP-based Web, email and FTP became critical entrenched applications. Till the mid-90s, ATM still offered considerable performance advantages over IP routers. Therefore, ATM was deployed as IP WAN backbones, and a lot of development went into the complex problem of internetworking IP and ATM (overviewed in the MPLS history section). Multi-Protocol Label Switching was then developed as a "hybrid" method of solving the IP-over-ATM problem, with a simple proposition: take the IP control plane and merge it with the ATM data plane. MPLS has been developed considerably since then, and ISPs are slowly moving away from ATM backbones to MPLS.

ATM uses virtual circuits (VCs) that are first routed using PNNI and established before communication can happen. Telephone networks use time-division multiplexing (TDM) and a multiplexing hierarchy (eg: T1, T3, OC1, OC3 etc). In ATM one can signal an ATM VC and operate at any rate. Since telephony networks are synchronous (periodic), ATM was labeled "asynchronous transfer mode". ATM uses cells (i.e. packets) to allow statistical multiplexing. ATM cell sizes are fixed at 53 bytes (48-bytes payload).

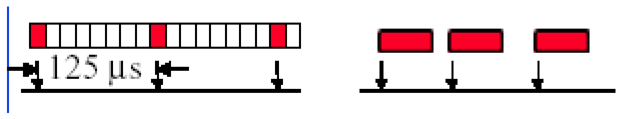

Figure 3: Synchronous vs Asynchronous Transfer Mode

ATM offers five service classes: constant bit rate (CBR), variable bit rate (VBR-rt and VBR-nrt), available bit rate (ABR), guaranteed frame rate (GFR) and unspecified bit rate (UBR). ATM offers adaptation layers for different applications to be mapped to lower level transmissions. ATM uses 32-bit labels (VCI and VPI) instead of addresses in its cells. ATM addresses are 20-byte long and are used in the signaling phase alone. During signaling, each switch on the path assigns an outgoing label from its local label space, and accepts the label assignment of the prior switch on the path for the inbound link. The ATM forwarding table entryhas four fields --

inbound label, inbound link, outbound link and outbound label. The label in the ATM header is hence swapped at every switch, unlike IP addresses that do not change en-route. ATM VC signaling involves specification of the traffic class and parameters, routing it through the network (using a scalable QoS-enabled link-state protocol, PNNI) and establishing a VC. Then cells flow through the path established. Traffic is policed, flow controlled or scheduled in the data-plane as per the service specification.

2.2.1 PNNI Routing in ATM networks

Private Network-to-Network Interface (PNNI) is the (somewhat odd) name of the routing protocol in ATM networks [2]. It is a QoS routing protocol designed to scale to very large network sizes, and supports hierarchical routing, multiple routing metrics and attributes. It uses link-state information, which is used by the source to formulate a source-route for signaling calls. Like other hierarchical link state protocols, it uses the notion of a peer group (aka area) where nodes within a peer group have visibility into the full topology of the peer group, and aggregated topology views of higher levels. A peer group leader (PGL) represents the group at a higher level. An example from the PNNI standard [2] is shown in Figure 4. Node A.1.1 sees the full topology of peer group A.1 (i.e. PG(A.1)), sees the high-level logical topology of PG(A), and sees the high-level logical interconnections between PG(A), PG(B) and PG(C).

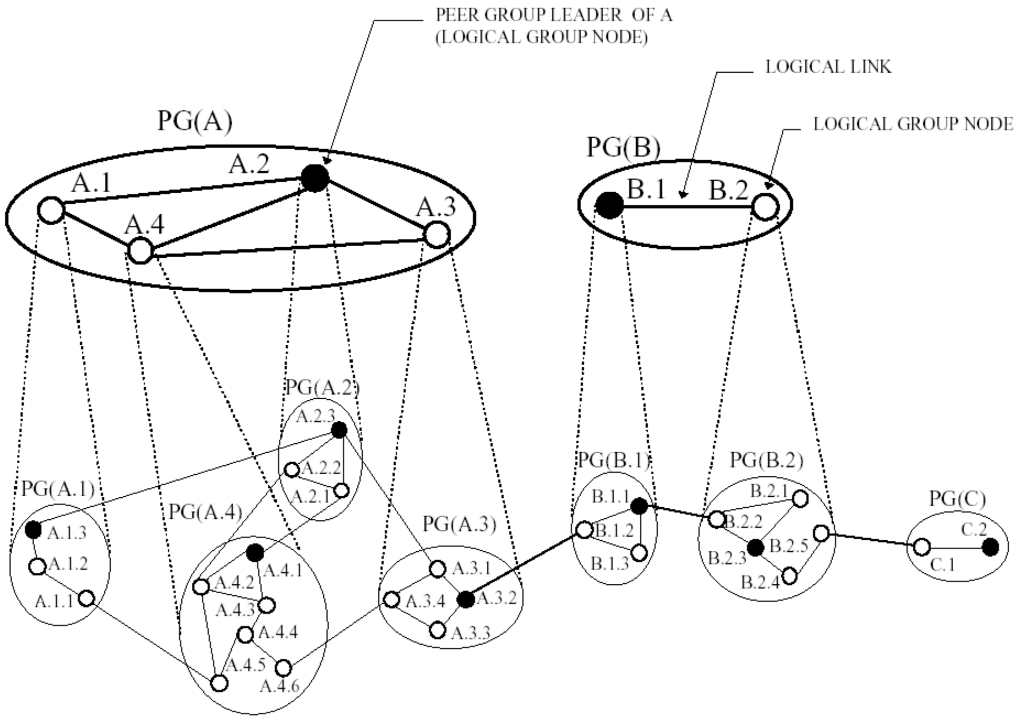

Figure 4: PNNI Hierarchical Routing: Addressing, Peer Groups, Leaders

PNNI encodes its source-route as a stack of Designated Transit Lists (DTLs) in the connection-setup (signaling) request. A simplified DTL processing case[4] is shown in Figure 5. Node A.1.1 wants to setup a VC to B.3. It chooses a route based upon QoS or policy routing considerations.

[4] This example is due to Prof. Raj Jain, Ohio State University

Its initial DTL-stack consists of three DTLs: [A.1.1 A.1.2], [A.1, A.2] and [A B] which specifies the full route. When the signaling enters peer group A.2, the entry node A.2.1 updates the first DTL as [A.2.1 A.2.3] specifying a source-route through the A.2 peer group. Similarly when the signaling enters peer group B, the entry node B.1 pops out the path prefix, and specifies a new DTL [B.1 B.2 B.3]. The receiver has enough information in the source route to acknowledge the signaling, and specify a reverse explicit route.

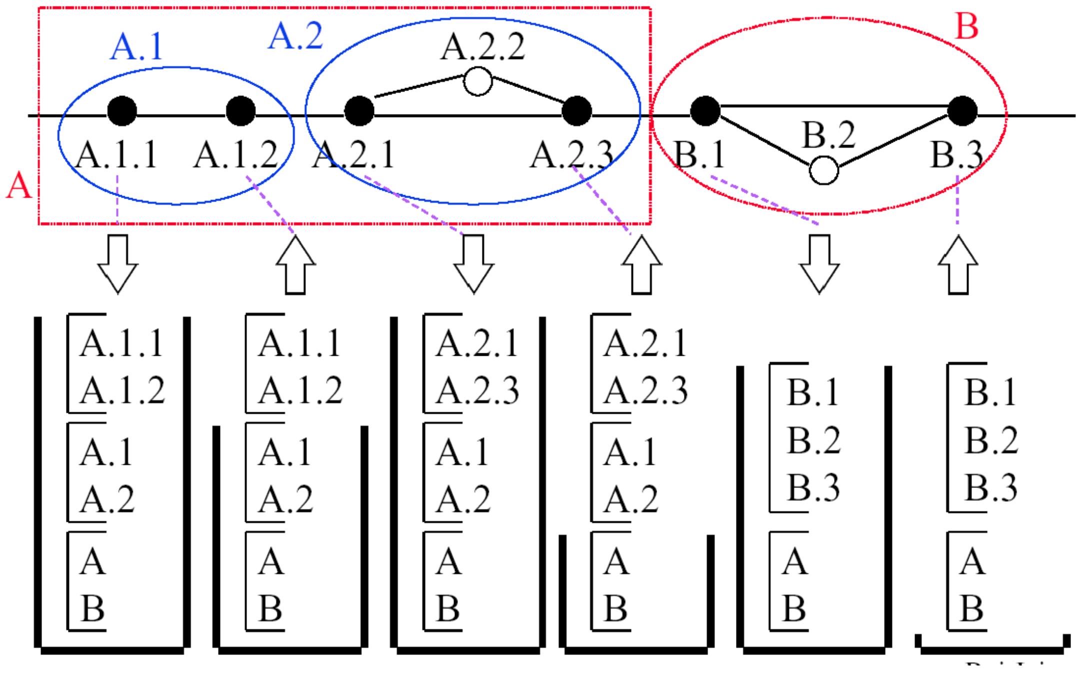

Figure 5: Designated Transit List (DTL) Path Specification and Processing

2.3 Signaled Routing Frameworks: MPLS

The overlay model using IP over ATM requires the management of two separate networks with different technologies (IP and ATM) resulting in increased operational complexity and cost. Some of the other issues with the overlay model are discussed in [4]. An example of the overlay model is shown in Figure 6.

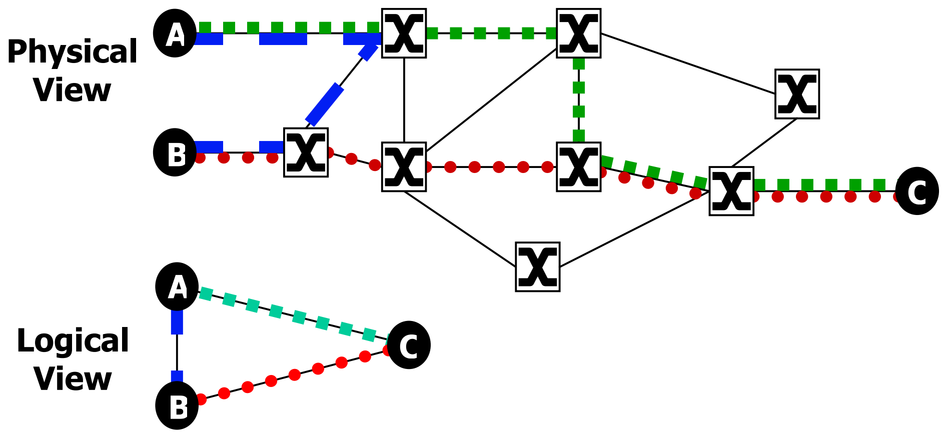

Figure 6: Overlay Model: Physical View vs Logical View

When IP is overlaid on ATM, the logical view seen by IP is given in the lower portion of Figure 6. However, this logical view has to be mapped into the physical topology. As discussed in the next section, this overlay model was very hard to realize.

MPLS was initially developed to increase the speed of IP forwarding. With increasing interface speeds it was considered hard to scale the IP forwarding capability per box proportionately. While advancements in electronics has obviated this motivation, the integration of the label switching paradigm into routers has made MPLS a powerful tool for TE in IP networks while alleviating the problems of an IP over ATM overlay. With advances in MPLS and router hardware technology MPLS based networks are being deployed more often [82], [89], [90]. The requirements for traffic engineering over MPLS are described in RFC2702 [85]. Extensions to RSVP to support instantiation of explicit LSP are discussed in RFC3209 [87]. Extensions to LDP, known as CR-LDP, to support explicit LSPs are presented in RFC3212 [88].

2.3.1 MPLS History

Multi-Protocol Label Switching (MPLS) was born out of the complex IP-over-ATM mapping problem and hence we briefly review the relevant issues. The Internet Protocol (IP) is in general a good internetworking solution based upon the "overlay" model, i.e., layering and mapping IP over heterogeneous link layers. IP is a very simple connectionless protocol and expects lower layers to minimally support a frame forwarding service. The IP-over-ATM encapsulation was specified in RFC 1483 [29] and is very simple. However, the essential protocols surrounding IP, and the IP subnet model place more requirements on lower layers.

In particular, the address resolution protocol (ARP) works best if the link layer supports broadcast. In non-broadcast multiple access (NBMA) media like ATM networks, an address resolution server (ARS) and virtual circuits (VCs) to the ARS are required (Eg: Classical IP over ATM, RFC 2225 [38]), creating a central point of failure and new scalability issues. The IP subnet model expects direct connectivity within a single subnet, and requires at least one router hop if multiple logical IP subnets are mapped to a physical network. The former issue imposes a costly full mesh VC connectivity requirement in an NBMA network. The latter issue led to inefficient multi-hop communications in a large ATM cloud even when a cut-through VC may have been possible (e.g. See [41]). The NHRP standard [41] attempted to have cut-through VCs at the expense of a complex next-hop resolution distributed procedure.

The Internet routing protocols (OSPF [44] and BGP-4[55]) need to maintain logical adjacencies (Hello protocol) with neighbors, generate Link State Advertisements (LSAs) for links, and synchronize their databases with neighbors. Routing protocols work best on point-to-point links. In VC-oriented networks like ATM, every logical adjacency requires a physical VC to be setup (and the costs of the VC to be borne, even if hello traffic is minimal). If the network-LSA encoding of the LSA is used [44], then full mesh connectivity is expected in the underlying VC-based network. Moreover a designated router (DR) and backup DR need to be configured and maintained for database synchronization purposes. Alternatively, OSPFv2 [44] proposes a point-to-multipoint subnet model that allows a single IP subnet to be mapped to a single NBMA, but without the restriction of full-mesh VC connectivity. However, it breaks the IP subnet requirement of connectivity within a subnet. BGP-4 requires the synchronization between i-BGP and e-BGP routers, and expects a full mesh between i-BGP nodes. This requirement again translates into a full mesh of physical VCs between i-BGP nodes that cannot be avoided!

Essentially all these problems stemmed from the *overlay* model used to map ATM to IP. When MPLS was designed, the goal was to take the IP control-plane (i.e. routing protocols) and *merge* it with the ATM forwarding plane. Observe that this merged model is different from overlaying a fully featured IP control- and data-plane over a fully featured ATM control- and data-plane. In other words, ATM switch hardware could be controlled by IP routing protocols. The mapping software was essentially a signaling or switch control protocol (e.g. LDP, RSVP, GSMP) that sets up a label-switched path based upon the routing information available from IP. Predecessors of MPLS included Ipsilon's IP Switching and Cisco's Tag Switching. Another motivation in doing such a mapping was to leverage the high-speed ATM switching hardware and building IP core networks. A fixed-length label match was expected to be much faster than a longest-prefix destination-address match. However, this performance gap was quickly erased with the development of new data-structures and ASICs for high-speed IP forwarding. As a result, MPLS was repositioned to serve as a key building block for traffic engineering.

2.3.2 Basics of an IP network with MPLS

At the ingress to an MPLS domain, label switching routers (LSRs) classify IP packets into forwarding equivalence classes (FECs) based on a variety of factors, including e.g. a combination of the information carried in the IP header of the packets and the local routing information maintained by the LSRs. An MPLS label is then pre-pended to each packet according to their forwarding equivalence classes. In a non-ATM/FR environment, the label is 32 bits long and contains a 20-bit label field, a 3-bit experimental field (formerly known as Class-of-Service or CoS field), a 1-bit label stack indicator and an 8-bit TTL field (see Figure 7) In an ATM (FR) environment, the label consists information encoded in the VCI/VPI (DLCI) field.

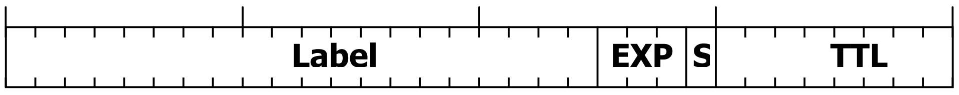

Figure 7: MPLS Header Fields

An MPLS capable router (an LSR) examines the label and possibly the experimental field and uses this information to make packet-forwarding decisions. An LSR makes forwarding decisions by using the label prepended to packets as the index into a local next hop label forwarding entry (NHLFE). The packet is then processed as specified in the NHLFE. The incoming label is replaced by an outgoing label, and the packet switched to the next LSR. This label-switching process is very similar to the label (VCI/VPI) swapping process in ATM networks. Before a packet leaves an MPLS domain, its MPLS label is removed. A Label Switched Path (LSP) is the path between an ingress LSRs and an egress LSRs through which a labeled packet traverses. The path of an explicit LSP is defined at the originating (ingress) node of the LSP. MPLS may use a signaling protocol such as RSVP-TE or CR-LDP to set up LSPs.

2.3.3 Current Limitations of MPLS

Currently, the TE link attributes in enhanced OSPF and IS-IS, and consequently in MPLS, are *area-scope* only. This limitation implies that detailed topology and traffic engineering details is available only inside an area of the network. The information, even in a summarized manner, is not available across areas. This limits the network size that operators can deploy MPLS for traffic engineering today. Sharing of information across areas will be required to establish end-to-end LSPs across multiple areas in large multi-area networks. Also since MPLS is a signaled protocol, it cannot be directly mapped to BGP to facilitate inter-AS TE capabilities. Therefore the problems of inter-area and inter-domain TE are largely open (though BGP-4 provides some simple hooks for inter-domain TE, see section 7).

3 QoS Routing vs. Traffic Engineering

Qos routing is an area related to TE, and it is important to distinguish the two. QoS routing refers to the problem of *finding a route to satisfy the QoS specification of a flow* [67], [68], [69]. QoS routing also includes some aspects of the signaling protocol that is used to set up the reservations (after admission control) at every hop. In particular, a QoS signaling protocol may support crank-back, i.e. the capability to regress to an intermediate node and try an alternate path if the original source-routed path is not available. PNNI surveyed earlier is an example of a QoS routing protocol.

Traffic engineering refers to a broader objective of performance improvement of existing networks that includes QoS routing issues, *and* efficient utilization of network resources. There are problems in QoS routing which attempt to include some view of overall network optimization, e.g. minimum interference routing, widest-shortest-path [70], [68], [76].

QoS-based routing extends the Internet routing paradigm in three basic ways. First, to support traffic using Intserv or Diffserv class of services, multiple paths between node pairs will have to be calculated. Routing updates (e.g. OSPF LSAs) can be extended to support such calculation. Second, traffic will be shifted from one path to another as soon as a "better" path is found. The traffic will be shifted even if the existing path can meet the service requirements of the existing traffic. To avoid this, mapping the traffic to paths should be de-coupled from path availability and path attractiveness. Third, today's QoS path routing algorithms do not support multi-path routing. If the best existing path cannot admit a new flow, the associated traffic cannot be forwarded even if QoS can be met by using multiple paths.

RFC-2676 [68] discusses the metrics required to support QoS, the extension to the OSPF link state advertisement mechanism to propagate updates of QoS metrics, and the modifications to the path selection to accommodate QoS requests. RFC-2676 does not consider a network with partial upgrades, i.e., establishing a route in a heterogeneous environment where some routers are QoS-capable and others are not. Support for QoS routing can be viewed as consisting of three major components:

1. Obtain the information needed to compute QoS paths and select a path capable of meeting the QoS requirements of a given request

2. Establish the path selected to accommodate a new request
3. Maintain the path assigned for use by a given request

The goal of the extensions is to improve performance for QoS flows (likelihood to be routed on a path capable of providing the requested QoS), with minimal impact on the existing OSPF protocol and its current implementation. Depending on the scope of the path selection process, this returned path could range from simply identifying the best next hop, i.e., a traditional hop-by-hop path selection model to specifying all intermediate nodes to the destination, i.e., an explicit route model (e.g. MPLS signaling). In addition to the problem of selecting a QoS path and possibly reserving the corresponding resources, one should note that the successful delivery of QoS guarantees requires that the packets of the associated "QoS flow" be forwarded on the selected path. This typically requires the installation of corresponding forwarding state in the router using mechanisms like RSVP, diff-serv PHBs or MPLS signaling protocols.

The extensions to LSAs considered in RFC 2676 include only available bandwidth and delay. In addition, path selection is itself limited to considering only bandwidth requirements. A simple extension to the RFC 2676 path selection algorithm allows directly incorporation for delay, assuming rate-based schedulers at bottlenecks [71]. In particular, the path selection algorithm selects paths capable of satisfying the bandwidth requirement of flows, while at the same time trying to minimize the amount of network resources that need to be allocated, i.e., minimize the number of hops used. Note that this does not fully capture the complete range of potential QoS requirements. For example, a delay-sensitive flow of an interactive application could be put on a path using a satellite link, if that link provided a direct path and had plenty of unused bandwidth. One approach to preventing such poor choices, is to assign delay-sensitive flows to a "policy" that would eliminate from the network all links with high propagation delay,

3.1 Path Selection Algorithms for QoS Routing

Optimal path selection under constraints is a general flow-theory optimization problem [73], [74] and in general an NP-complete problem [69]. There are two broad classes of work in this context. The first class attempts heuristics for the general problems [69]. The second class defines special cases of the general problem that are of practical interest in data networks, and where polynomial solutions exist [69], [75]. For example, computing paths with a minimum bandwidth while minimizing hop-count is an important problem identified in RFC-2676 [68].

Chen and Narstedt [69] survey a large number of unicast and multicast QoS routing algorithms. In particular, they divide unicast algorithms into “basic” and “composite” algorithms. *Basic* algorithms have either a link- or path-optimization requirement, or a link-/path-constraint.

For some QoS metrics like residual bandwidth and residual buffer space, the state of a path is determined by the state of the bottleneck link. An example of *link-optimization* routing is to find a path that has the largest bandwidth on the bottleneck (e.g. widest-path [76]). An example of *link-constrained* routing is to find a path whose bottleneck bandwidth is above a required value. Both these problems can be solved with a modified version of Dijkstra algorithm [78] or Bellman-Ford [77] algorithm.

For other QoS metrics like delay, delay jitter and additive cost metrics, the state of a path is determined by the combined state over all links on the path. Similar to the link-based problems, one can define path-optimization and path-constrained routing problems. An example of a *path-optimization* problem is to find the least-cost (or min-delay) path, where "cost" or "delay" refers to path-cost or path-delay. An example of path-constrained routing is to find a path whose delay is bounded by a required value. Again both the problems can be solved by variants of the Dijkstra or Bellman-Ford algorithms.

Many composite problems can be defined combining the above basic classes of problems e.g., link-constrained, link-optimization (LCLO) routing or path-constrained, path-optimization (PCPO) routing. It turns out that while many of these problems have polynomial algorithms, two important composite problems are NP-complete: path-constrained, path-optimization (PCPO), and multi-path-constrained (MPC) routing. An example of PCPO is delay-constrained least-cost routing, and an example of MPC is delay- *and* delay-jitter-constrained routing.

A lot of work in QoS routing algorithms has revolved around either finding heuristics for the NP-complete class, or solving composite routing variants that have polynomial costs in an optimal fashion. Recently, work in hierarchical QoS routing has concluded that hierarchy and topology aggregation introduces inaccuracy in routing information and has a negative effect on QoS routing [75]. Impact of routing protocol overhead, dynamic load shift and stale information has been studied by Shaikh, Rexford, Shin ([79], [80]).

3.2 Path Management vs. QoS Routing

It is important to notice that consistent delivery of QoS guarantees implies stability of the data path. In particular, while it is possible that after a path is first selected, network conditions change and result in the appearance of "better" paths, such changes should be prevented from unnecessarily affecting existing paths (as happens in today's hop-by-hop approaches). In particular, switching over to a new (and better) path should be limited to specific conditions, e.g., when the initial selection turns out to be inadequate or extremely "expensive". This aspect is beyond the scope of QoS routing and belongs to the realm of path management [72].

4 Connectionless Traffic Engineering

In this section we discuss traffic engineering (TE) with intra-domain connectionless routing algorithms such as OSPF. We also discuss the multi-path approach for TE in the connectionless framework. Finally, we discuss the work related to fast restoration techniques in this framework. It is hard to achieve a range of TE capabilities with DV protocols such as RIP and EIGRP because of the limited view of the network at each router.

4.1 Traffic Engineering by Optimizing OSPF weights

The shortest path nature of OSPF is the main limitation in achieving TE goals using OSPF. OSPF routes traffic on shortest paths based on the advertised link weights. As a result, the link along the shortest path between the two nodes may become congested while the links on longer paths may remain idle. Link weights are often set statically, inversely proportional to link bandwidth, based on the Cisco recommendation [11]. In this section we examine the problem of load-balancing, one of TE goals, using OSPF.

4.1.1 Adaptive Routing Vs Load Balancing by Optimizing OSPF weights

One of the earlier approaches to achieve load-balancing in an OSPF network was by *dynamically adapting* link weights to reflect the local traffic conditions on a link [27], [37], [39]. This is called adaptive routing or traffic-sensitive routing. However, adapting link weights to *local* traffic conditions leads to frequent route changes and is unstable (See [6], [15] for stability analysis). Independent local decisions are made by the routers to set and advertise the link weights. The assumption here is, the routers do not have any knowledge of the traffic load on distant links and therefore cannot optimize traffic allocation. However, assuming that an estimate of the entire traffic demand is known, *globally* optimum link weights can be found under quasi-static traffic assumptions. This is the approach taken in [23], [24], [25], [63]. On a side note, issues related to estimating the demands (on a source-destination level) by monitoring the traffic at the ingress routers have been studied in [22]. In all these cases, a *central* network management entity is assumed to compute the ``optimal" link weights and deploy in the network using a network management protocol such as SNMP. These link weights lead to the desired routing under OSPF. This approach alleviates issues related to stability and excess overhead associated with traffic-sensitive routing.

However, finding optimal link weights for a given traffic demand is a NP-Hard problem [24]. A local search algorithm has been developed by Fortz and Thorup [23][24][25] to obtain the optimum link weights for a given demand matrix. The objective is to minimize a convex, piece-wise linear, increasing (with offered load on a link) function over all the links in the network. For the proposed AT&T backbone network under the projected traffic demands, optimal link weight setting in OSPF led to a performance that is close to *optimal general routing* [69]. *Optimal general routing* refers to routing where traffic may be optimally split among various paths between a source and destination. *Optimal routing* can be achieved in MPLS framework by carefully setting up Label Switched Paths (LSPs). In [63], Tao et al have formulated the *packet loss rate* in the network as the optimization metric. An expression for packet loss rate has been derived in terms of the link parameters, such as bandwidth and buffer space, and the parameters of the traffic demands using a GI/M/1/K queueing model. They have developed a fast recursive random search (RRS) algorithm so that the link-weight optimization may be performed frequently (to take into account changing demands). The RRS technique has been shown to take 50-90% fewer iterations as compared to the local search scheme in [23] to find a ``good" link weight setting.

Changing OSPF link weights leads to a transient period when the routers are advertising new link weights and re-computing the shortest paths. Change in routes may lead to out-of-order arrival of TCP packets, formation of transient loops etc. Hence, changing OSPF link weights too frequently may not be such a good idea. Thus, in [25], Fortz et al have investigated performance improvements in OSPF with only a few weight changes. They have developed a heuristic search algorithm for finding link weight setting that lead to improved performance with minimal link weight changes. It may be noted that there is a trade-off in optimality by reducing the number and frequency of link weight changes.

Feldman et al [21] have developed a simulation tool NetScope to characterize some dynamic effects of the new OSPF weight settings. Administrators can locate a heavily loaded link in the network, identify the traffic demands that flow though it, and change the configuration to reduce the congestion. An interesting question regarding TE by optimizing OSPF weights is: *How good is the best OSPF routing as compared to optimal general routing*. This question has been answered in [24]. In [24] authors have constructed an example with n^3 nodes such that the average utilization seen by the flow under OSPF routing (with unit link weights on all links) is $\Omega(n)$ higher than that seen in *optimal general routing*. In [40], authors have categorized into destination-based routing (such as OSPF) and source-destination based routing (such as MPLS with only one LSP between a source-destination pair). They have found upper and lower bounds on the ratio of performance for these cases under the Max Flow criterion. The lower bound is found to be $\Omega(\log(n))$ whereas $\Theta(n)$ represents the upper bound.

Two factors contribute to this difference in performance between OSPF and MPLS. First, is the single path nature of OSPF that allows routing traffic on the shortest path to destination. The feature of Equal Cost Multi Path (ECMP) was provided to achieve some traffic splitting. Using ECMP, OSPF allows the traffic is distributed equally among various next hops of the equal cost paths between a source and a destination [44]. This is useful in distributing the load to several shortest paths. However, the splitting of load by ECMP is not optimal as shown in [24]. Figure 8 illustrates the traffic splitting in ECMP. Secondly, due to the nature of shortest path, if link (i,j) $\in$ SP(s,d) then link(i,j) $\in$ SP(i,d). Hence, only a subset of paths can be used for any given link weight setting. Using MPLS, it is possible to arbitrarily setup a path between a source and destination using explicit signaling. Also, multiple paths may be setup and traffic may be split among these paths with some granularity.

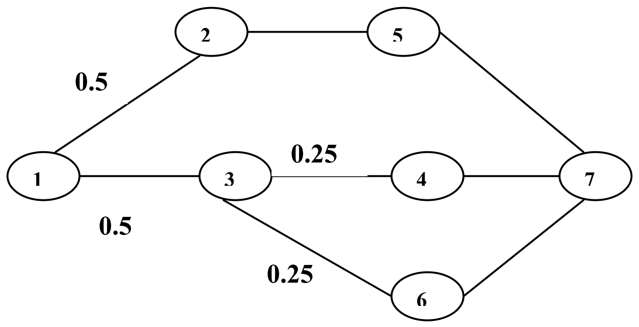

Figure 8: Traffic splitting in OSPF when all links have unit weight

4.2 Multi-Path Routing

Multi-path routing aims to exploit the resources of the underlying physical network by providing multiple paths between source-destination pairs. Multi-path routing has a potential to aggregate bandwidth on various paths, allowing a network to support data transfer rates higher than what is possible with any one path [56]. The work in the area of multi-path routing has focused mainly on extending intra-domain routing algorithms (both RIP and OSPF) for multi-path support [9], [45], [59]. There are two aspects of a multi-path routing algorithm: *computation* of multiple loop-free paths and *traffic splitting* among these multiple paths. Extensive work has been done in both these areas. Distributed multi-path routing algorithms can be viewed as an extension of hop-by-hop routing algorithms. Centralized multi-path algorithms can be used in the MPLS framework to influence establishment of LPSs. However, traffic splitting algorithms may be used to the traffic among multiple paths in both the cases.

4.2.1 Traffic Splitting Among Multiple Paths

Optimal splitting with ECMP has been researched in OSPF-Optimized Multi Path (OMP) [58]. OSPF-OMP uses ECMP, but instead of depending upon weight assignments, it samples traffic load information and floods it via opaque LSAs. This information is used to change local load splitting decisions.

Gallager [26] developed a distributed computation framework for multi-path routing that guarantees minimum average delays, assuming stationary inputs and links. (Quasi-static case in not studied and might have implications to the update cost and noisier measurements of marginal link delays and node flows). Also, the routing algorithm needs to start with an initial loop free set. While the proof assumes infinitely divisible traffic, it remains to be seen how this will operate with coarser granularity flows.

The optimal routing algorithm of [26] is extremely difficult if not impossible to implement in real networks because of its stationary or quasi-static assumptions and the requirement of the knowledge of global constants. Several other algorithms have been proposed in literature [5], [8], [51], [52] that improves this minimum delay algorithm of [26]. Extensions of the minimum delay algorithm to handle topological changes are proposed in [5], [8] which use distributed routing techniques developed in [43]. An improved technique for measuring the marginal delays is presented in [8] while [5] uses second derivatives to increase the convergence rate of the original minimum delay algorithm. In [59] a simpler algorithm is proposed which can achieve near-optimal routing. The impact of granularity on network and routing behavior is studied in [54]. The authors noticed that while finer granularity improved network performance, this does not carry over when routing updates are made at smaller time scales. Finer granularities also imply higher classification/processing at nodes/ingress that results in more expensive packet forwarding.

Another approach is to associate a linear or piece-wise linear penalty function with every link in the network and the objective is to minimize the penalty for all links in the network. Penalty function may be chosen to approximate the packet drop rate or delay using M/M/1 queueing model. Then the traffic split may be obtained by solving the linear-programming (LP) problem (referred to as *optimal general routing* in [23]).

In [62], Wang et. al. have formulated a linear optimization problem to minimize the maximum of link utilization. Optimal traffic split can be obtained by solving this LP problem. However, they have used the dual formulation to obtain the link weights that would lead to optimal routing under the assumption of arbitrary traffic splitting across multiple ECMPs. This is an important result because for a linear objective function, any optimal routing can be implemented using the existing shortest path routing. One may note that since this is a centralized approach, it also assumes knowledge of *global* demand matrix and quasi-static traffic.

Simple local heuristics at the source/intermediate routers may also lead to a substantially improved performance as compared to the default OSPF routing. A study of various traffic splitting techniques, tradeoff of computational complexity and performance is an area open to research.

While multi-path routing allows multiple paths between a source and destination, how packets are split on these paths is an important issue. Sending TCP packets from a single flow on multiple paths with different round-trip-time (RTT) would lead to out-of-order packet arrivals and hence, performance degradation. Three main approaches modulo-N hash, hash-threshold and highest random weight have been described and studied in RFC-2991 [57]. The most commonly used scheme is hash-threshold. Performance of this scheme has been studied in greater detail in RFC-2992 [30]. However, if hashing is done on the source-destination IP address using a CRC16, the traffic distribution can only be done with a coarse granularity [60]. Using a 32-bit CRC to hash on source-destination IP address and port number leads to a finer granularity. However, using TCP port numbers at every hop is expensive. In [60], authors propose to treat UDP and TCP packets in a different fashion. Out-of-order packet arrival is not too expensive for UDP packets. However, they propose a modified version of hash-threshold scheme. They propose to generate and insert in the packet a random key based on the source-destination IP addresses and port numbers. This is done at the source/ingress router. Within the core network the key is used to determine the next-hop at each router. They propose to use the 13-bit fragment packet offset field to store the hash key (if fragment packet offset field is zero, i.e. the packet is un-fragmented). They also propose to use the TOS or DSFIELD bit to indicate whether the packet is a UDP or TCP packet. This eliminates the need to read packet header at each hop and facilitates faster forwarding. However, the trade-off is out-of-order arrival of fragmented packets.

4.2.2 Multi-path Computation Algorithms

So far we have focused on schemes for traffic splitting given multiple paths. Now we discuss distributed and centralized algorithms for multi-path computation. Centralized algorithms may facilitate in establishing LSPs in the MPLS framework.

In [45], authors present Multiple Path Algorithm (MPA) that can be implemented as an extension to OSPF. MPA finds only a subset of paths that satisfy a condition for loop-freeness. However, it does not find all loop-free paths to a destination. A router only considers paths with next-hop such that the weight of shortest path from next-hop to destination is less than the weight of the shortest path from router to destination.

In [61], authors extend the DV algorithms to compute loop-free multi-paths at every instant. They propose MDVA that is free from count-to-infinity problem, provides multiple next-hop choices for each destination and the routing graphs implied by these paths are always loop-free. The MDVA algorithm has been designed around a set of loop-free invariant conditions that ensure instantaneous loop-freedom and prevents count-to-infinity.

A standard Depth-First Search (DFS) [14] may be used to find all paths between a source and destination. However, number of paths between a source and destination grows exponentially with the size of the network. Hence, storing all paths in the forwarding table is memory-expensive. Two key questions arise *how many paths are needed* and *how to compute these paths*.

These issues have been addressed in and a near-optimal scheme developed in [47]. Schemes for computing k-shortest paths have been developed in [7], [19], [42]. Various algorithms for computing k-shortest paths have been compared in [7].

In order to implement multi-path routing in the connectionless framework a consistency criterion must be met or some form of source routing must be used to forward packets. A mathematical formulation of a multi-path forwarding framework and extensions to both LS, DV have been proposed in [9]. Authors have demonstrated that forwarding in *suffix matched* multi-path protocols can be efficiently implemented using a path identifier in the packet header. The router memory needed is linear in K, the number of paths between a source and destination. They have also showed that distributed multi-path routing algorithms and ranked k-shortest paths algorithms yield *suffix matched* multi-paths. However, this work proposes a label-switched realization of multi-path algorithms that are hard to map to current routing protocols like OSPF and BGP.

4.3 Restoration methods in OSPF/IS-IS

Link state based routing protocols like OSPF and IS-IS dominate the current Internet. Primary reason for this is that in a LS routing protocol each router has the complete map of the network will allows for TE extensions, QoS/constraint-based routing etc. The convergence time of OSPF/IS-IS after a failure or change is believed to be order of seconds. Routing loops, out-of-order packets, packet-loss etc. may occur during this convergence period. One of the goals of TE is to improve the reliability of network operations. In this section, we discuss the work done in the area of restoration methods for LS routing protocols. The work done in this area focuses on either proposing algorithms to communicate new paths between routers or modify existing OSPF/IS-IS for faster convergence after change in network state.

In [65] authors have investigated reason for slow convergence of LS protocols such as IS-IS/OSPF and provide recommendations to improve the convergence time. They propose sub-second *Hello Interval*, usage of incremental shortest path computation algorithms, give priority to LS packets' propagation than shortest path computation. They suggest setting *Hello* timers based on link bandwidth and propagation delay rather than arbitrarily. Also, prioritizing *Hello* packets in front of data packets may ensure stability with small *Hello Interval* under congestion. In [66], authors have investigated stability and overhead associated with TE extensions to OSPF and the impact of sub-second *Hello Intervals*. Their findings show that using sub-second *Hello* timers considerably improves the convergence times of OSPF without significantly adding to the processor loads.

In [46], authors propose a vector-metric algorithm for restoration after multiple link failures. The algorithm uses a vector link state metric instead of scalar link state metric. This allows routers to stop using the metric for the failed link and route traffic on the shortest *restoration path*. The routers attached to the failed link immediately know of its failure. They compute the path to the destinations reached by the failed link. This is the *restoration path*. The information of the failed link is only conveyed to the routers on the shortest *restoration path*. This limits the amount of computational and bandwidth overhead of flooding LSAs after a link failure. It also reduces the recovery time, thus reducing the transient period of possible routing loops.

In [64], authors present an algorithm to route traffic without causing loops in the event of a link failure. The forwarding decisions are based on both the previous and current view of the network. They use the concept of a *restoration path* that is similar to [46]. However, they provide an algorithm for packet forwarding in such cases instead of using the vector metrics.

Issues associated with fast recovery after link failures in an important research topic. With virtual connection being set up on the top of Layer-3 routing protocols such as OSPF, it is faster to restore connectivity after link/router failure at the Layer-3. The recommendations in [65] provide a basis for fast link/router failure detection/convergence in routing protocols such as IS-IS/OSPF.

5 Signaled (Connection-oriented) Traffic Engineering

An *overlay* network is the term used to describe interconnecting routers over an underlying virtual circuit (VC) network. The VC network, such as ATM and Frame Relay, provides VC connectivity between routers that are located at the edges of a virtual-circuit cloud. In this model, two routers that are connected through a VC see a direct adjacency between themselves independent of the physical route taken by the VC through the underlying network. Thus, the overlay model essentially decouples the logical topology that routers see from the physical topology that the ATM or Frame Relay manages. The overlay model based on VCs enables a network operator to employ traffic engineering concepts to perform path optimization by re-configuring or rearranging the VCs so that traffic on a congested or sub-optimal physical links can be re-routed to a less congested or more optimal ones. Traffic engineering using VCs relies on key components [84], [85], [82] like Constraint based routing [67], enhanced link state IGP, connection admission control. These are briefly reviewed below.

5.1 Constraint-based Routing

Constraint-based routing [17], [85] refers to a class of routing systems that compute routes through a network subject to satisfaction of a set of constraints and requirements.

In order to control the path of VCs effectively, each VC can be assigned one or more attributes. These attributes are considered in computing the path for the VC. Such attributes include:

Attribute	Meaning
Bandwidth	Value required on all links in the defined path for the VC
Hop Count	Maximum number of nodes the path can traverse
Setup Priority	Determines the order in which this VC will get a resource if multiple VCs compete for it
Hold priority	Determines whether an established VC should be preempted for this VC
Resource class (color)	An administratively specified attribute that decides the resources this VC can use

Adaptability	If the VC can be switched to a more optimal path when one becomes available
Resilience	Whether to reroute the VC when the current path fails

Other attributes can pertain to diversity and protection aspects on links used. Some of these are:

Shared Risk Link Group	Maximally diverse paths shared between a common source-destination.
Protection	Links can be 1+1, 1:1, 1:N configurable.
Include Resources	Nodes/links which must be included in the path
Exclude Resources	Nodes/links which must not be included in the path

The above attributes as constraints and requirements may be imposed by the network itself or by administrative policies. Constraints may include bandwidth, hop count, delay, and policy instruments such as resource class attributes. Such constraints impose restrictions on the solution space of the routing function. Since constraint based routing considers more than network topology in computing routes, it may be possible to find a longer but lightly loaded path better than the heavily loaded shortest path.

Constraint based routing can be online or offline. With online routing, edge nodes in the network may compute for VCs at any time. In the most general setting, constraint-based routing may also seek to optimize overall network performance while minimizing costs [84]. An offline server computes paths for all VCs periodically and then configures VCs along these paths.

Unlike QoS routing (for example, See [67]) which generally addresses the issue of routing individual traffic flows to satisfy prescribed flow based QoS requirements subject to network resource availability, constraint-based routing is applicable to traffic aggregates as well as flows and may be subject to a wide variety of constraints which may include policy restrictions.

5.2 Enhanced Link State IGPs

Constraint based routing requires additional link attributes beyond the usual connectivity information. Common link attributes [36] include maximum bandwidth, maximum reservable bandwidth, unreserved (available) bandwidth per priority and resource class (color). Other attributes include protection type on a link and Shared Risk Link Group (SRLG) - a set of identifiers referring to entities e.g. conduits, fibers whose failure can cause the link to fail [86]. These attributes help constraint based routing choose links with appropriate protection and help in diverse path computation. Enhanced link state IGPs flood link state updates if there is a change in the link attributes e.g. available bandwidth parameter. Excessive flooding is avoided by imposing a ceiling on flooding frequency and/or ensuring link state updates are done only when there is a *significant* change in bandwidth.

5.3 Solving Traffic Engineering Problems with MPLS

Traffic engineering with MPLS requires the components of constraint based routing and an enhanced IGP discussed in the last section. With MPLS when an enhanced IGP builds LSR's forwarding table, it takes into account LSPs originated by the LSR, so that LSPs can be used to carry traffic. As discussed earlier, IGPs using shortest path to forward traffic attempt to conserve resources but can lead to congestion. This can be due to different shortest paths overlapping at some link or the traffic from a source to a destination exceeding the capacity of the shortest path. MPLS helps address the above problems and more as described below.

Constraint based routing, along with some form of connection admission control, avoids placing too many LSPs on any link, thus avoiding one of the problems. Similarly, if the traffic between two routers exceeds the capacity of any single path, then multiple LSPs can be set up between them. The traffic is split between these based on specified or derived load ratios. For example, the ratios may be proportional to the bandwidths of the LSPs. Further, such LSPs can be placed on different physical paths to ensure more even distribution of load. This also allows for graceful degradation in case one of the paths fail.

MPLS allows enforcement of some administrative policies in online path computation. For example, resource color can be assigned to LSPs and links to achieve a degree of desired LSP placement. [82] suggests an example where regional LSPs are to be kept from traversing inter-region links. To enforce this scheme, all regional links may be colored green, and all inter-region links colored red. Regional LSPs are then constrained to use only green links. If an operator chooses, paths for LSPS may be determined offline, possibly based on global optimization and other administrative policies considerations. This allows network administrators [90] to control traffic paths precisely.

In terms of support for network planning, an advantage of MPLS is the ability to gather per-LSP statistics that can be used to provide point-to-point traffic matrix. In particular, such point-to-point traffic matrix provides valuable historical data for longer term network planning. Such data can be used for capacity planning. While capacity planning is the long-term solution to meet growing traffic demands, in the medium term an operator has to work within the bounds of the given capacities. During this period the traffic is growing and changing distribution. A powerful tool is the ability offered by MPLS to adjust the bandwidth of existing LSPs in the network. This allows operators to adjust LSP bandwidths to reflect the changing traffic distribution.

The priority feature in MPLS may be used in interesting ways for traffic engineering. For example, [82] suggests a LSP that carries large amount of traffic to be given higher priority. This allows the LSP to more likely take an optimal path resulting in better resource utilization from a global perspective. The priority feature may be used in other ways as well, for example, to offer different grades of service.

Another feature that can be useful to an operator is the ability to re-optimize the path of an LSP. Basically at the time an LSP is set up it may not obtain the best path between the source and the destination due to existing paths. With time, however, resources may be added or freed up that cause better paths to become available. A network operator may want to switch the LSP to a better path when it becomes available. Re-optimization should be done with care in a controlled manner as it may introduce instability.

Rerouting of a LSP is also desirable when a failure occurs along its path. Rerouting can be end-to-end *(global repair)*, where the whole path is rerouted from ingress to egress, or *localized*, where only a section of the LSP is rerouted to avoid a failed link or node. In the case of failure, it may be even desirable to reroute LSPs regardless of their bandwidth and other constraints. Rerouting and more generally survivability in MPLS networks is an important topic [81], [83]. Exploiting the reroute capabilities of MPLS requires the use of signaling to set up LSPs [89]. The general concepts and techniques for survivability in MPLS networks is discussed in the next section.

MPLS also offers two other key benefits. A Differentiated Services (Diffserv) over MPLS implementation [91] allows the operator to combine traffic engineering with LSPs while supporting differentiated services. This may be achieved in one way by setting up different LSPs for different classes of traffic. In an alternative scheme a single LSP may carry traffic belonging to different classes. A detailed discussion of Differentiated Services on MPLS is beyond the scope of this paper. The other key aspect exploited about MPLS is not its capability of traffic engineering, but of mapping services like Virtual Private Networks to labels. Thus, core nodes transport these as LSPs, agnostic of the service being carried inside. Discussion on this aspect of MPLS is outside the scope of this paper.

6 Survivability in MPLS Based Networks

Network survivability refers to the capability of a network to maintain service continuity in the presence of faults. This can be accomplished by promptly recovering from network impairments and maintaining the required QoS for existing services after recovery. Survivability has become an issue of great concern within the Internet community due to the increasing demands to carry mission critical traffic, real-time traffic, and other high priority traffic over the Internet [4]. Rerouting is traditionally used at the IP layer to restore service following link and node outages. Rerouting at the IP layer occurs after a period of routing convergence that may require seconds to minutes to complete.

To support advanced survivability requirements, path-oriented technologies such as MPLS can be used to enhance the survivability of IP networks in a potentially cost effective manner. Because MPLS is path-oriented it can potentially provide faster and more predictable protection and restoration capabilities than conventional hop by hop routed IP systems. The advantages of path oriented technologies such as MPLS for IP restoration becomes even more evident when class based protection and restoration capabilities are required [3].

Broadly, protection types for MPLS networks can be categorized as *link protection, node protection, path protection, and segment protection*. In all these cases a protection or backup LSP path needs to be computed disjoint from the portion of the primary LSP that is being protected. The nodes at the ends of the portion of the LSP being protected are sometimes referred as *points of repair*. The backup LSP is set up between the points of repair. When the portion of the LSP spans a link it is called link protection. Similarly when it spans a node and its adjacent links it is called node repair. When it spans a segment it is segment repair and when it spans the whole path of the LSP it is path repair. Note that path or segment repair protect against all possible link and node failures along the path or segment. Additionally, since the path selection is end-to-end, path protection might be more efficient in terms of resource usage than link or node protection. However, path protection may be slower than link and node protection in general.

Restoration speed in MPLS depends on the protection technique used e.g. link, node, path-based. Restoration time depends on the time to detect the fault and notify the point of repair [81], [83]. Once the notification is received, restoration time depends on the switchover time to a backup LSP. Switchover time is a function of whether the backup LSP is computed and established after fault, pre-computed but established on fault, or pre-computed and pre-established.

Below we first review the fault detection and notification schemes and then briefly discuss the mechanisms to achieve link, node, and path repairs in MPLS.

6.1 Fault Detection and Notification

Detection of a fault: The transport network carrying a link may have fast mechanisms (e.g. SONET alarms) to detect a fault like a fiber cut. Such transport alarms may be leveraged by the two routers at the ends of the failed link to detect the fault in a few milliseconds. In absence of transport alarms, local faults can only be detected via control plane mechanisms e.g. routing and signaling. Protocols like OSPF and RSVP have their respective hello mechanisms that may be used to detect a fault. These are expected to take more time. Control plane techniques may also help to detect a failed node.

Fault Notification: Once a failure is detected, a node adjacent to the failure can proceed with link or node protection on LSPs that have requested such. LSPs that have requested path or segment repair, and the fault is not adjacent to the point of repair, there is a step involved of notifying these nodes of the fault. Fault notification in a MPLS network may employ one of several techniques:

- **Advertisement via link-state routing** OSPF advertises to others about a link failure and consequently the point of repair comes to know about the fault. This would typically take tens of seconds.
- **Signaling mechanisms:** This notification occurs at a LSP level. Along each LSP signaling messages are sent both upstream and downstream to notify other nodes along the LSP of the fault. These messages may be sent hop-by-hop along the nodes or directly addressed to the point of repair (Notification message in [53]).

In-band techniques of fault notification, similar to *OAM* in ATM are currently not supported in MPLS.

6.2 Link repair

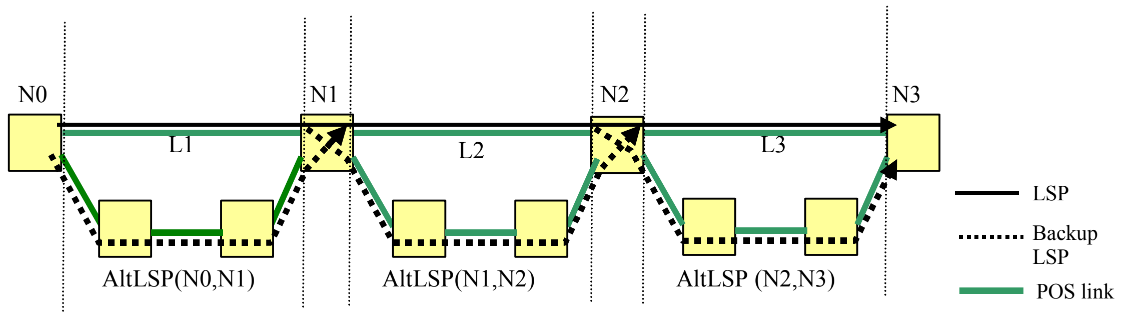

Figure 9: Depicts link protection. Primary LSP shown (in solid black line) over links L1, L2, and L3. For every hop of the LSP, there is pre-established a backup LSP denoted as Alt_LSP(N_{i-1} ,N_i) and depicted as black dashed lines.

Once the failure of a link is detected by adjacent nodes, a switch over is performed for the failed LSP segment to the backup LSP. Consider a fault on link L2 (second link). Internally node N1, when it learns about the fault, needs to unicast it to all incoming links. The link that has the LSP coming in, has to be *pre-programmed* that on failure of L2 to use the backup Alt_LSP(N1,N2) which is on a different outgoing link. Node N2 now receives traffic from the backup Alt_LSP (N1,N2). It merges the packets coming on it to the primary LSP. The restoration time in this scheme is a function of the number of LSPs to be switched over. For the number of LSPs in low thousands a router can be engineered to restore under 50 ms. There are different schemes for local protection being offered by vendors today [83]. These schemes differ in the details of the scheme and how the backup LSPs are set up and labels for them allocated. Figure 10 offers an insight into the label allocation problem for backup LSPs.

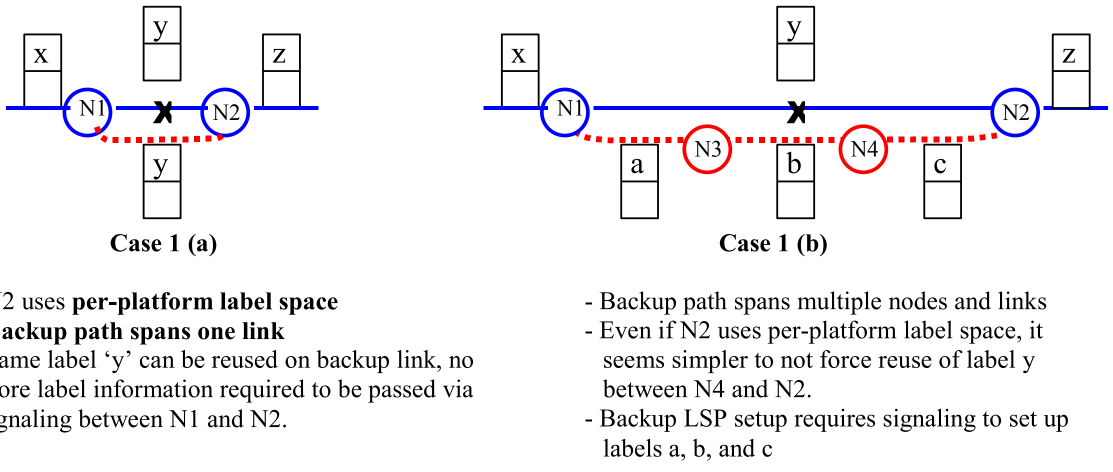
x
y
z
N1
N2
y
Case 1 (a)
- N2 uses **per-platform label space**
- Backup path spans one link
- Same label 'y' can be reused on backup link, no more label information required to be passed via signaling between N1 and N2.
x
y
z
N1
N2
N3
N4
a
b
c
Case 1 (b)
- Backup path spans multiple nodes and links
- Even if N2 uses per-platform label space, it seems simpler to not force reuse of label y between N4 and N2.
- Backup LSP setup requires signaling to set up labels a, b, and c

Figure 10: Label Allocation Problem for Backup LSPs

6.2.1 Aggregating Local Repair via Bypass Tunnel LSP

Link protection requires a backup segment for each LSP for each link that it traverses. Bypass tunnel suggests setting up a single backup LSP to protect a link. An LSP that traverses the link is protected by tunneling its backup via a bypass tunnel. Label stacking is used to differentiate between the backup LSPs.

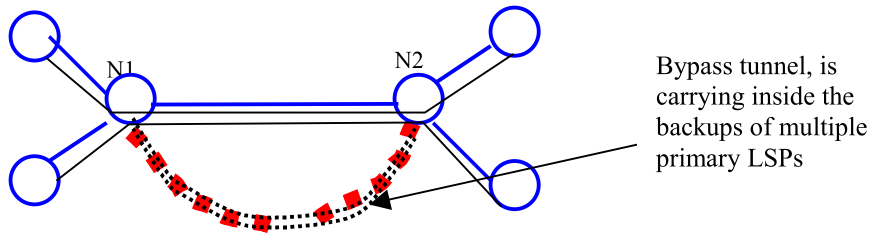
N1
N2
Bypass tunnel, is carrying inside the backups of multiple primary LSPs

Figure 11: Protection aggregation via bypass tunnel. A bypass tunnel is used to protect multiple LSPs traversing the same link. In the figure, two LSPs, depicted in thin lines, share the same link from N_1 to N_2. A bypass tunnel, depicted in thick dashed line, can be used to protect both LSPs. Label stacking is used to differentiate between the protection LSPs.

The bypass tunnel for a link can be set up dynamically or via manual configuration. If a backup tunnel is configured for every link, the logical topology with backup tunnels exactly parallels the link topology. When a LSP requesting link protection is signaled, a backup LSP is created on the bypass tunnel between the same two nodes. In the event of failure of the link, the LSP is rerouted to the next hop using the bypass tunnel. A new label corresponding to the bypass tunnel is pushed on. The tail end node on receiving the packet, first pops the label corresponding to the bypass tunnel. The exposed label then determines the protection LSP. The protection LSP is merged with the remaining downstream portion of the working LSP.

Note that the bypass tunnel mechanism, as proposed in [83] does not stress on maintaining QoS. This is adequate if the primary motivation for link protection is maintaining connectivity and not QoS. It works well if complemented with the strategy of end nodes of the LSP detecting the failure and doing a path-switchover. It is also possible to conceive several enhancements over the basic scheme. For example, bandwidth may be assigned to a backup tunnel. Similarly, class based differentiation may be supported by setting up a backup tunnel for each class.

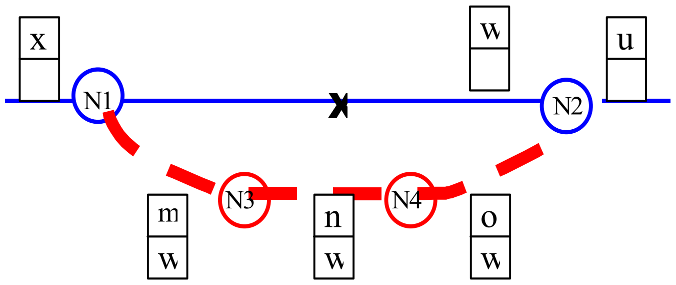

Figure 12: Label allocation with Bypass Tunnels: Example labels used on the primary LSP and the two deep label stack used on the bypass tunnel. It shows the case when per-platform label space is used on merge node N2. Thus, it is possible that for a given LSP, that the last label used along N2, denoted by 'w', be reused across the bypass tunnel.

The figure above shows that if the merge node (N2) is using platform label space, then the label necessary for the back up tunnel is the label already assigned by the merge node along the primary. If merge node does not support per-platform label space, then N2 and N1 negotiate another label different from ‘w’ using signaling across the bypass tunnel.

6.3 Node Repair

Tolerating node failures along with link failures, requires a scheme as depicted below.

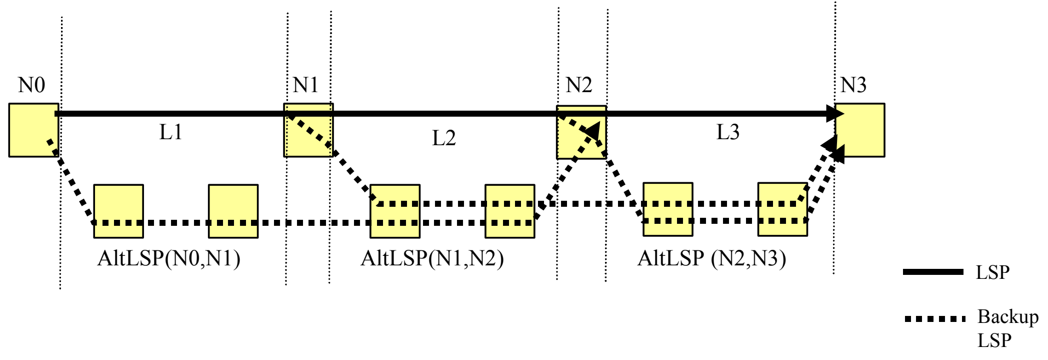

Figure 13: Similar to link repair, except two-hop backup LSPs set up to circumvent a node and its adjacent links.

A backup LSP (or an aggregated bypass tunnel) is created to bypass a node being protected against and meets up two nodes downstream. At the last hop, a backup LSP can only protect the last link. In the event of failure of any but the last link, the LSP is rerouted to the two hop downstream node using the appropriate two-hop backup LSP. Failure of an intermediate node triggers the same behavior. If the last link fails then the LSP is rerouted via the one hop backup LSP.

If the merge node uses platform label space, and a bypass tunnel is used for protection, then the label necessary for the back up along the tunnel is the label already assigned by the merge node along the primary. This requires the node at the head of the tunnel to learn the label used by the node that is two hops downstream. Signaling schemes that record the labels used on the LSP provide such a mechanism. If the merge node does not support platform label space, then N2 and N1 need to negotiate another label using signaling along the bypass tunnel.

6.4 Path repair

In the case of path repair, the whole LSP is protected by a backup LSP. The backup LSP is set up between the same source and the destination nodes as the primary. The figure below depicts path repair. The alternate LSP is the one shown in dashed pink. In this case using bypass tunnels is not feasible since the two ends can be arbitrarily far apart. The number of bypass tunnels that need to be pre-configured becomes too large. Thus, backup LSP is always used instead of tunnels and the backup LSP is set up via signaling.

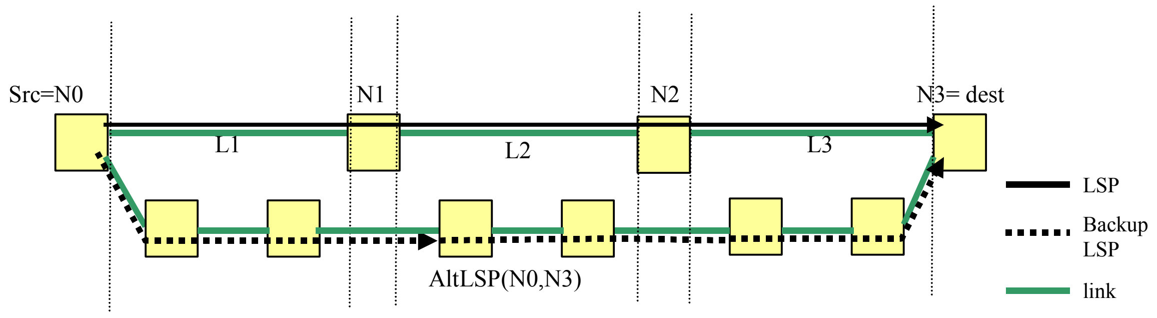

Figure 14: Depicts path repair on a primary LSP shown in solid line over links L1, L2, and L3. The backup LSP is depicted in dashed line.

Restoration time is a function of whether the backup LSP is computed and established after fault, pre-computed but established on fault, or pre-computed and pre-established.

7 Inter-Domain Traffic Engineering

In the inter-domain area, the IRTF is considering requirements documents for a future inter-domain protocol. Traffic engineering appears as an important problem in both the key proposals [16], [32]. Inter-domain TE work has revolved around multi-homed Autonomous System's (AS), in-bound/out-bound load-balancing between adjacent AS's using BGP attributes (e.g. Multi Exit Discriminator (MED), LOCAL_PREF, AS_PATH) [32], provider-selection and multi-homing issues considered with IPv6 development [48], [31], or map-distribution based approaches (NIMROD) [8].

BGP provides some simple capabilities for TE between AS neighbors [32]. The MED attribute can be used by an AS to inform its neighbor of a preferred connection (among multiple physical connections) for *inbound* traffic to a particular address prefix. Usually it is used by ISPs on the request of their multi-homed customers. Lately, it is also being used between ISPs. The LOCAL_PREF attribute is used locally within the AS to prefer an *outbound* direction for a chosen address prefix, AS or exit router. Recent work by Feamster et al [20], and our group [34] show that it is possible to automatically tune the LOCAL_PREF parameters of ``hot-prefixes'' to control outbound traffic subject to a range of policy constraints. The AS-PATH attribute may also be used to achieve TE. AS_PATH may be ``stuffed'' or ``padded'' with additional instances of the same AS number to increase its length and expect lower amount of inbound traffic from the neighbor AS to whom it is announced. However, this may lead to a large overhead if done too often.

Another way to achieve some TE is to subvert the BGP-CIDR address aggregation process. In particular an AS may extract more-specifics, or de-aggregate it and re-advertise the more-specifics to other AS's. The longest-prefix match rule in IP forwarding will lead to a different route for the more specific address. However, this is achieved at the expense of larger number of entries in forwarding tables. A way to avoid subverting CIDR aggregation (shown in our recent work [34]), in the limited case of multi-homed stub AS, is by mapping the inbound load-balancing problem to an address management problem. In particular, we re-map chosen flows or aggregates to different inbound links (or inbound providers) by choosing a public address from that provider's allocation for the chosen flows. The dynamic mapping between the chosen public address and the private address of the flow is done through dynamic NAT. In other words, we avoid BGP completely for this particular scenario. Alternatively, AS neighbors may agree on BGP community attributes [55] (that are not re-advertised) to specify traffic engineering. In summary, BGP (like OSPF) is yet another example of an Internet routing protocol that was *not* originally designed for traffic engineering, but some limited TE capabilities can be achieved by careful parameter tuning.

The NIMROD and IPv6/GSE proposals explicitly attempt to achieve some TE objectives. NIMROD [8] is a hierarchical, map distribution-based architecture that allows explicit source-based path choice, through a signaled or datagram (i.e. IP source-route) method. Due to the use of maps (link-states), NIMROD is incompatible with current BGP routing and may require signaling to setup explicit paths. IPv6 provides a routing option for provider selection, and elegant auto-configuration methods that easily accommodate site multi-homing. The provider-selection option was intended to allow choice of providers (potentially remote) in the forwarding path; which is not the same as a full route encoding. The IPv6 strict/loose source routing options are similar to that in IPv4, i.e. they enumerate the path (or partial path) and do not parsimoniously encode it.

Mike O'Dell's 8+8/GSE [48] proposes to break the IPv6 address into a locator (routing group) and an end-system designator (ESD) part. Unlike IPv4 and IPv6 addresses where the IP addresses are not modified en-route, 8+8/GSE proposes that the locator field could change after crossing autonomous systems (e.g. providers). This would facilitate multi-homing, change of providers, easy site renumbering and be a vehicle for TE capabilities like provider-selection, path-selection. GSE was proposed mainly in the context of renumbering, but there have been some private communications about how it could be adapted for traffic engineering. 8+8/GSE has been controversial in the IETF and has been tabled for further discussion, and hence it is unclear whether it may re-emerge.

8 Summary and Conclusions

There are two major approaches to traffic engineering (TE): connectionless and connection-oriented. Connectionless *routing* has been broadly adopted in the Internet, so there has been a lot of attention in extending it for traffic engineering approaches. However, due to the coupling between data-plane and control-plane in this model, changes in one affect the other leading to a need for widespread upgrades. Therefore most of the current connectionless approaches use indirect (or parameter tuning) methods for achieving TE objectives. Connection-oriented (or more specifically path-oriented or signaled) approaches allow a more direct realization of TE objectives. The TE field in the Internet received a lot of attention from the early-mid 1990s starting from the IP-over-ATM mapping problem. MPLS was proposed as an alternative to merge the IP control-plane and ATM data-planes. In particular, MPLS data-plane is de-coupled from the control-plane, and the data-plane provides efficient, opaque encapsulation and label-stacking capabilities. MPLS therefore provides flexibility for using arbitrary control plane algorithms to achieve TE objectives ranging from QoS routing, multi-path routing/traffic splitting, load balancing, path protection and fast re-route. We surveyed a range of work in both connection-oriented and connectionless models, in both intra-domain and inter-domain and it is clear that today the connection-oriented model is more advanced in terms of development and deployment.

9 Open areas of research

The Internet today provides only a single path between any pair of hosts that fundamentally limits the throughput achievable between them. For example, dramatically faster large file transfer or higher frame-rate video would be possible if applications could expect that multiple transport level sessions would be mapped to different paths in the network, and they have control over splitting traffic between these sessions.

While the late 1990s has led to widespread interest in development and adoption of TE capabilities, this has been *limited* to be within ISP domains (more specifically areas, not domains). It is clear the connection-oriented TE is being developed at a tremendous pace, especially since MPLS and G-MPLS provides a common basis for their evolution. In particular, a range of TE objectives ranging from QoS routing, multi-path routing/traffic splitting, load balancing, path protection and fast re-route can be directly addressed by a connection-oriented MPLS/G-MPLS based approach.

It is fair to observe that *connectionless TE and inter-domain TE* are relatively under-developed, and advances in this area are critical before the vision of expecting multiple end-to-end paths becomes a reality. In particular, the coupling between data- and control-planes of connectionless protocols must be addressed, and the forklift upgrade requirements must be relaxed. Overlay networks (eg: [1]) may also provide an interim experimental testbed for such advances. We believe that the development and widespread of such inter-domain TE capabilities, and breaking of the access bottleneck would lead to innovations in new high-bandwidth applications and will in turn drive the demand for bandwidth (and intra-domain TE) in ISP domains.

Bibliography

[1] D. Andersen et al, ``Resilient Overlay Networks," *ACM SOSP*, October 2001

[2] ATM Forum, Private Network-Network Interface Specification Version 1.0 (PNNI 1.0), *International Standard*, *af-pnni-0055.000*, March 1996

[3] D. Awduche et al, ``Overview and Principles of Internet Traffic Engineering," *IETF Internet Draft* draft-ietf-tewg-principles-02.txt, Work-in-progress, Jan 2002.

[4] D. Awduche, ``MPLS and traffic engineering in IP networks," *IEEE Communications Magazine*, Vol. 37, No. 12, pp. 42-47, 1999.

[5] D. P. Bertsekas, E. M. Gafni, and R. G. Gallager, ``Second derivative algorithms for minimum delay distributed routing in networks," *IEEE Transactions on Communications*,] vol. 32, no. 8, pp. 911-919, August 1984.

[6] D. P. Bertsekas, ``Dynamic Models of Shortest Path Routing Algorithms for Communication Networks with Multiple Destinations," *Proceedings of 1979 IEEE Conference on Decision and Control*, Ft. Lauderdale, FL, pp. 127-133, Dec. 1979.

[7] A.W. Brander, M.C. Sinclair, ``A Comparative Study of K-shortest Path Algorithms," In *Proceedings of 11th UK Performance Engineering Workshop*, Liverpool, pp.370-379, September 1995.

[8] C. G. Cassandras, M. V. Abidi and D. Towsley, ``Distributed routing with on-line marginal delay estimation," *IEEE Transactions on Communications*, vol. 38, no. 3, pp. 348-359, March 1990.

[9] I. Castineyra, N. Chiappa, M. Steenstrup, ``The Nimrod Routing Architecture," *IETF RFC 1992*, August 1996.

[10] J. Chen, P.Druschel, D.Subramanian, ``An Efficient Multipath Forwarding Method," in *INFOCOM'98*, March, 1998.

[11] Cisco, Configuring OSPF, 1997.

[12] Douglas Comer, ``Internetworking with TCP/IP Vol.1: Principles, Protocols, and Architecture," 4th Edition, Vol. 1. *Prentice Hall*, 2000.

[13] D. Coppersmith and and S. Winograd, `` Matrix Multiplication via Arithmetic Progression," *ACM Symp. on Theory of Computing (STOC)*, 1987, pp.1-6.

[14] T.H. Cormen, et. al., ``Introduction to Algorithms", *The MIT Press, McGraw-Hill Book Company*, Second Edition, 2001.

[15] Zheng Wang, Jon Crowcroft, ``Analysis of Shortest-Path Routing algorithms in a Dynamic Network Environment," *Computer Communication Review*, Vol. 22, no. 2, pp.63-71, 1992.

[16] E. Davies et al, ``Future Domain Routing Requirements," *IRTF Routing Research Draft*draft-davies-fdr-reqs-01.txt, Work-in-progress, July 2001.

[17] A. Elwalid, et al, ``MATE: MPLS Adaptive Traffic Engineering," *INFOCOM'01*, April, 2001

[18] End-to-End Mailing List Thread, ``Traffic engineering considered harmful," June 2001.

[19] D. Eppstein, ``Finding the k shortest paths," in *Proceedings of 35th Symposium on Foundations of Computer Science*, IEEE, pp.154-165, 1994.

[20] N. Feamster, J. Borkenhagen, J. Rexford, ``Controlling the Impact of BGP Policy Changes on IP Traffic," *Technical Report, AT\&T Research Labs*, 2001.

[21] A. Feldman, A. Greenberg, C. Lund, N. Reingold, J. Rexford, ``NetScope: Traffic Engineering for IP Networks," *IEEE Network Magazine*, March 2000.

[22] A. Feldmann, Albert G. Greenberg, Carsten Lund, Nick Reingold, Jennifer Rexford and Fred True, ``Deriving traffic demands for operational IP networks: methodology and experience", In *Proceedings of SIGCOMM*, pp. 257-270, 2000.

[23] B. Fortz, M. Thorup, ``Internet Traffic Engineering by Optimizing OSPF Weights, in *Proceedings of the INFOCOM 2000*, pp. 519-528, 2000.

[24] B. Fortz, M. Thorup, ``Increasing Internet Capacity Using Local Search," *Preprint*, 2000.

[25] B. Fortz, M. Thorup, ``Optimizing OSPF/IS-IS Weights in a Changing World," *IEEE JSAC*, to appear.

[26] R. G. Gallager, ``A Minimum Delay Routing Algorithm Using Distributed Computation," *IEEE Transactions on Communications*, Vol. COM-25, No. 1, January 1977. pp. 73-85.

[27] D. W. Glazer and C. Tropper, ``A New Metric for Dynamic Routing Algorithms," *IEEE Transactions on Communications*, Vol. 38 No.3 March 1990.

[28] J. Gray, T. McNeill, ``SNA multiple-system networking," *IBM systems Journal*, Vol. 18, No. 2, 1979.

[29] J. Heinanen, "Multiprotocol Encapsulation over ATM Adaptation Layer 5," IETF RFC 1483, July 1993

[30] C. Hopps, ``Analysis of an Equal-Cost Multi-Path Algorithm," *IETF RFC 2992*, 2000.

[31] C. Huitema, ``IPv6: The New Internet Protocol," *Prentice Hall*, Second Edition, 1998.

[32] G.Huston, ``Commentary on Inter-Domain Routing in the Internet," *IRTF Routing Research Draft,* draft-iab-bgparch-02, Work-in-progress, Sept 2001.

[33] V. Jacobson, SIGCOMM 2001 Keynote Address.

[34] S. Kalyanaraman, H. T. Kaur, T. Ye, S. Yadav, M. Doshi, A. Gandhi, ``Load Balancing in BGP environment using Online Simulation and Dynamic NAT," at *ISMA Workshop*, by CAIDA, Dec. 2001.

[35] K. Kar, M. Kodialam, T. V. Lakshman, ``Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications," *IEEE JSAC*, Vol. 18, No. 12, December 2000.

[36] D. Katz, D. Yeung, K. Kompella, ``Traffic Engineering Extensions to OSPF," *Internet draft,* draft-katz-yeung-ospf-traffic-06.txt, Work-in-progress, Jan 2002.

[37] Atul Khanna and John Zinky, ``The Revised ARPANET Routing Metric," *Proceedings of the ACM SIGCOMM*, pp. 45-56, 1989.

[38] M. Laubach and J. Halpern, "Classical IP and ARP over ATM", RFC 2225, April 1998.

[39] D. S. Lee, G. Ramamurthy, A. Sakamoto and B. Sengupta ``Performance analysis of a threshold-based dynamic routing algorithm," *Proceedings of the Fourteenth International Teletraffic Congress*, ITC-14, 1994.

[40] D.H. Lorenz, A. Orda, D. Raz, Y. Shavitt, ``How good can IP routing be?" *DIMACS Technical Report 2001-17*, May 2001.

[41] J. Luciani et al ``NMBA Next Hop Resolution Protocol (NHRP)", *IETF RFC 2332*, April 1998.

[42] E.Q.V. Martins, M.M.B. Pascoal and J.L.E. Santos, ``The K Shortest Paths Problem," *Research Report, CISUC*, June 1998.

[43] P. M. Merlin and A. Segall, ``A failsafe distributed routing protocol," *IEEE Transactions on Communications*, vol. 27, no. 9, pp. 1280-1287, September 1979.

[44] J. Moy, ``OSPF Version 2," *IETF RFC 2328*, April 1998

[45] P. Narvaez, K. Y. Siu, ``Efficient Algorithms for Multi-Path Link State Routing," *ISCOM'99*, Kaohsiung, Taiwan, 1999.

[46] P. Narvaez, L.Y. Siu, H. Y. Tzeng, ``Local Restoration Algorithm for Link-State Routing Protocols," in *Proceedings of the IEEE ICCCN*, 1999.

[47] S. Nelakuditi, Z-L. Zhang, ``On Selection of Paths for Multi path Routing," In *Proceedings of IWQoS*, 2001.

[48] M. O'Dell, ``GSE -- an alternate addressing architecture for IPv6," *Expired Internet Draft*, 1997.

[49] F. Roman, ``Shortest Path Problem is Not Harder than Matrix Multiplication," *Information Processing Letters*, Vol. 11, 1980, 134-136.

[50] E. Rosen et al, ``Multi-Protocol Label Switching Architecture," *IETF RFC 3031*, January 2001.

[51] A. Segall, ``The modeling of adaptive routing in data-communication networks," *IEEE Transactions on Communications*, vol. 25, no. 1, pp. 85-95, January 1977.

[52] A. Segall and M. Sidi, ``A failsafe distributed protocol for minimum delay routing," *IEEE Transactions on Communications*, vol. 29, no.5, pp. 689-695, May 1981.

[53] P. Ashwood-Smith et al, "Generalized MPLS - Signaling Functional Description", *Internet Draft*, draft-ietf-mpls-generalized-signaling-07.txt, 2002.

[54] A. Sridharan, S. Bhattacharyya, C. Diot, R. Guerin, J. Jetcheva and N. Taft. ``On the impact of aggregation on the performance of traffic aware routing," *Technical Report*, University of Pennsylvania, Philadelphia, PA, June 2000.

[55] J. Stewart III, ``BGP-4 Inter-domain routing in the Internet," *Addison Wesley*, 1999.

[56] H. Suzuki and F. A. Tobagi, ``Fat bandwidth reservation scheme with multi-link and multi-path routing in ATM networks," In *Proceedings of IEEE INFOOCOM*, 1992.

[57] D. Thaler, C. Hopps, ``Multipath Issues in Unicast and Multicast Next-Hop Selection," *IETF RFC 2991*, 2000.

[58] C. Villamizar, ``OSPF Optimized Multipath (OSPF-OMP)," *Expired Internet Draft*, 1999. Available: http://www.ietf.org/proceedings/99mar/I-D/draft-ietf-ospf-omp-02.txt

[59] S. Vutukury and J.J. Garcia-Luna-Aceves, `` A Simple Approximation to Minimum-Delay Routing," *SIGCOMM '99*, September, 1999.

[60] S. Vutukury and J.J. Garcia-Luna-Aceves, `` A Traffic Engineering Approach based on Minimum-Delay Routing," *Proceedings of IEEE ICCCN 2000*, Las Vegas, Nevada, USA, October 2000.

[61] S. Vutukury and J.J. Garcia-Luna-Aceves, ``MDVA: A Distance-Vector Multipath Routing Protocol," In *Proceedings of the INFOCOM*, 2001.

[62] Z. Wang, Y. Wang, L. Zhang, ``Internet Traffic Engineering without Full Mesh Overlaying," *INFOCOM'01*, April 2001.

[63] T. Ye, Hema T. Kaur, S. Kalyanaraman, K. S. Vastola, S. Yadav, ``Dynamic Optimization of OSPF Weights using Online Simulation," Preprint available from *http://www.ecse.rpi.edu/Homepages/shivkuma/research/papers-rpi.html*, 2002.

[64] R. Rabbat, K. Y. Siu, ``Restoration Methods for Traffic Engineered networks with Loop-Free routing Guarantees," in *Proceedings of the International Conference on Communications (ICC),* June 2001.

[65] C. Alaettinoglu, V. Jacobson, H. Yu, ``Towards Milli-Second IGP convergence," *Internet Draft*, draft-alaettinoglu-ISIS-convergence-00, 2000.

[66] Anindya Basu, Jon G. Rickle, ``Stability Issues in OSPF Routing," in *Proceedings of SIGCOMM*, 2001.

[67] E. Crawley et al, “A Framework for QoS-based Routing in the Internet” *IETF RFC 2386*, August 1998 .
[68] G. Apostolopoulos et al, “QoS Routing Mechanisms and OSPF Extensions,” *IETF RFC 2676*, August 1999.
[69] S. Chen, K. Nahrstedt, An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions, IEEE Network Magazine, Special Issue on Transmission and Distribution of Digital Video, Vol. 12, No. 6, November-December 1998, pp. 64-79.
[70] K. Kar, M. Kodialam, T. V. Lakshman, Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications , *IEEE Journal on Selected Areas in Communications, Vol. 18, No. 12, December 2000.*
[71] Y. Goto, M. Ohta and K. Araki, "Path QoS Collection for Stable Hop-by-Hop QoS Routing", Proc. INET '97, June, 1997.
[72] R. Guerin, S. Kamat, and S. Herzog. QoS Path Management with RSVP. In Proceedings of the 2nd IEEE Global Internet Mini-Conference, pages 1914-1918, Phoenix, AZ, November 1997.
[73] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, “Network Flows,” Prentice-Hall, NJ, 1993.
[74] Cormen, Leiserson and Rivest, “Introduction to Algorithms,” MIT Press, Cambridge MA, 1990.
[75] R. A. Guerin, "QoS Routing in Networks with Inaccurate Information: Theory and Algorithms," *IEEE/ACM Transactions on Networking*, Vol. 7, No. 3, June 1999. http://citeseer.nj.nec.com/280763.html
[76] Z. Wang and J. Crowcroft, “QoS Routing for Supporting Resource Reservation, IEEE JSAC, September 1996.
[77] R.E. Bellman, “Dynamic Programming,” Princeton University Press, Princeton, NJ, 1957.
[78] E. Dijkstra, “A Note on Two Problems in Connection with Graphs,” Numerische Mathematik, Vol 1, 1959, pp. 269-271.
[79] A. Shaikh, J. Rexford, and K. Shin, "Load-sensitive routing of long-lived IP flows," Proc. ACM SIGCOMM, September 1999, pp. 215-226
[80] A. Shaikh, J. Rexford, and K. Shin, "Evaluating the impact of stale link state on quality-of-service routing," IEEE/ACM Transactions on Networking, April 2001.
[81] V. Sharma et. al., "Framework for MPLS Based Recovery", *Internet Draft*, draft-ietf-mpls-recovery-frmwrk-03.txt, 2001..
[82] X. Xiao, A. Hannan, B. Bailey, L. Ni, "Traffic Engineering with MPLS in the Internet," IEEE Network magazine, Mar. 2000.
[83] P. Pan et al, “Fast Reroute Techniques in RSVP-TE”, *Internet Draft*, draft-pan-rsvp-fastreroute-00.txt, 2002.
[84] M. K. Girish, B. Zhiu, and J-Q. Hu, “Formulation of the Traffic Engineering Problems in MPLS Based IP Networks”, Proceedings of the Fifth IEEE Symposium on Computers and Communications (ISCC 2000).
[85] D. Awduche et al, “Requirements for Traffic Engineering Over MPLS”, *IETF RFC 2676*, Sept 1999.

[86] K. Kompella et al, “OSPF Extensions in Support of Generalized MPLS”, *Internet Draft*, draft-ietf-ccamp-ospf-gmpls-extensions-04.txt, 2002.

[87] D. Awduche et al, “RSVP-TE: Extensions to RSVP for LSP Tunnels”, *IETF RFC 3209*, Dec 2001.

[88] B. Jamoussi et al, “Constraint-Based LSP Setup using LDP”, *IETF RFC 3212*, Jan 2002.

[89] V. Springer, C. Pierantozzi, and J. Boyle, “Level3 MPLS Protocol Architecture”, expired Internet Draft available at, http://www.watersprings.org/links/mlr/id/draft-springer-te-level3bcp-00.txt, 2000.

[90] C. Liljenstolpe, “Offline Traffic Engineering in a Large ISP Setting”, Internet Draft, draft-liljenstolpe-tewg-cwbcp-01.txt, 2002.

[91] F. Le Faucheur et al, “MPLS Support of Differentiated Services”, *Internet Draft*, draft-ietf-mpls-diff-ext-09.txt, 2001.