MPLS Advantages for Traffic Engineering

George Swallow, Cisco Systems

ABSTRACT This article discusses the architectural aspects of MPLS which enable it to address IP traffic management. Specific MPLS arhcitectural features discussed are separation of control and forwarding, the label stack, multiple control planes, and integrated IP and constraint-based routing. The article then discusses how these features address network scalability, simplify network service integration, offer integrated recovery, and simplify network management. Scalability is addressed through integrated routing enabling a natural assignment of traffic to the appropriate traffic engineering tunnels without requiring special mechanisms for loop prevention. Change is greatly reduced. The label stack enables an effective means for local tunnel repair providing fast restoration. Feedback through the routing system permits fast and intelligent reaction to topology changes. Service integration is simplified through a unified QoS paradigm which makes it simple for services to request QoS and have it mapped through to traffic engineering.

The traffic engineering problem can be defined as an optimization problem. Given a fixed topology and a fixed source-destination matrix of traffic to be carried, what routing of flows offers the best overall performance? For a specific network the objective function must represent the goals of the network administration. But generally the question is, "how does one make the most effective use of the available bandwidth?" or "how can the traffic be laid into the topology such that all the traffic is delivered without unreasonable delay?" The key to any solution is having the ability to place the flows; the finer the control on this placement, the closer one can get to the optimum. However, too fine a grain, such as individual Transmission Control Protocol (TCP) sessions, presents a scalability problem. Experience has shown that for many networks, aggregating traffic at the city-pair level allows one to get close to the optimum and still have a tractable number of flows to engineer.

Normal IP unicast routing is destination based. At each router a packet is forwarded based solely on its destination address. This happens regardless of the source of the packet. At each router, flows to a destination are thus aggregated. From a routing scalability perspective, this is very desirable. Maintaining routes on a source-destination simply isn't scalable.

From a traffic engineering perspective, however, normal unicast routing can be highly sub-optimal. As in any packing problem, a few large flows are more difficult to load balance on a given topology, than is the same traffic presented as many smaller flows. Adjusting routing metrics can only act on the aggregated flows. Multipath routing is of some use in creating and routing finer grained flows, but its usefulness is limited. To get reasonably close to the optimum, additional mechanisms are needed to manage IP flows. These are the ability to aggregate traffic into appropriate-sized flows, and then to explicitly route those flows through the network.

THE OVERLAY SOLUTION

A common technique used among large IP service providers is to use a layer 2 network to manage the network bandwidth. The IP backbone is connected by a complete mesh of virtual circuits (VCs). This serves to thwart the normal aggregation that occurs hop by hop in an IP backbone with destination-based routing. The aggregate flow from each router to each of the other routers can be individually routed through the layer 2 topology. A limited number of routers in each city are connected to the mesh and serve as aggregation points for the traffic. With this strategy, very effective traffic engineering can be achieved.

There are drawbacks to this approach, however. Issues of scalability, manageability, and cost arise.

A simple link failure can mean dozens of VCs going down, forcing the IP routing protocols into a major reconvergence. Further, IP has no idea of the metrics associated with the physical topology. When dealing with less than the full mesh, IP will have to choose routes based on artificial metrics. The alternate routes so chosen may be extremely suboptimal.

Further, link state protocols have particularly poor scalability in fully meshed topologies. Suppose there are n routers in a mesh. If one router fails, then each of its n-1 neighbors will see its link to that router fail and generate a link state update. Each router will flood its update on n-2 links. Each of the n-1 routers will then receive n-2 new updates. They will in turn reflood these on n-3 links. Therefore, the total number of updates is $O(n^3)$. In network protocols, things of $O(n^2)$ are generally considered to be unscalable.

The overlay solution is more costly. Aside from the obvious extra network devices, rack space, and power, there is usually a requirement for additional personnel. Also, there are complications in network management. Problem resolution must be coordinated between the layer 2 network and the IP network.

A more scalable, manageable, and cost-effective solution is needed. Multiprotocol label lwitching (MPLS) is particularly useful in this regard.

KEY MPLS ARCHITECTURAL FEATURES

In this section we discuss some of the architectural features of MPLS which make it particularly useful for traffic engineering. One important goal of the MPLS architecture is to combine the scalability and flexibility of routing with the performance, quality of service (QoS), and traffic management of layer 2 switching. A result of this is to make capabilities available to the IP layer, which traditionally have only existed at layer 2. The primary aspects of MPLS that enable this are separation of control and forwarding, a unified forwarding paradigm, and the label stack. We now discuss each of these. Later we'll discuss how these capabilities of MPLS can be exploited for IP traffic engineering.

SEPARATION OF CONTROL AND FORWARDING

The key architectural principle of MPLS is a clean separation of control and forwarding. This separation enables effective

 0163-6804/99/$10.00 © 1999 IEEE

service integration including QoS transparency between service layers.

MPLS has a simple label-switching paradigm that applies to all traffic. Various applications using MPLS can directly manipulate label bindings to effect the desired behavior of the forwarding elements. Label semantics are completely up to the control plane. A label may represent a fine-grained *micro* flow, or a coarse-grained *macro* flow. It can represent unicast or multicast traffic. Labels can be used to implicitly route packets as in native IP forwarding or they can be used to explicitly route packet in the same way as an asynchronous transfer mode (ATM) VC.

Forwarding Component

The forwarding component of MPLS is designed to be simple and efficient. While this is motivated by a desire to allow the forwarding to occur in hardware, it also makes the forwarding algorithm independent of control module.

As a result, all of the control modules share a single QoS paradigm. Label lookup and the *experimental bits* determine both the output queue, and drop priority. This is certainly not unique to MPLS. What is unique is that the same QoS mechanisms are invoked regardless of which control plane assigned the labels. The QoS mechanisms available to IP traffic engineering, as well as any layered services such as virtual private networks, are the same. No awkward, possibly incomplete QoS translation occurs as it often does when mapping IP onto ATM.

Label Stack

Unlike frame relay which has a single label, the data link connection identifier (DLCI), or ATM which has two, the virtual path identifier/virtual channel identifier (VPI/VCI), MPLS allows an arbitrary number of labels. The labels are simply stacked. A field in the label encapsulation indicates whether this label is the bottom of the stack. There are three label operations, push, pop, and swap. The swap operation is analogous to VCI or VPI switching. It represents the movement of the packet through a device at a single control level. The push operation adds a new label to the top of the stack. Usually this represents a new control element taking action on the packet. The pop operation removes one label from the stack. This usually represents the return of the previous control element.

Each label is 20 bits; when encapsulated it is carried in 32 bits. The remaining bits are, 3 experimental bits – generally assumed at the time of this writing to be reserved for class of service (CoS)/QoS — 8 bits of time-to-live (TTL), and the stack bit.

The power of the label stack is that it allows multiple control components to act on a packet. Further, this can occur with little or no coordination between the control planes, making it simple to combine services in useful ways. For example, traffic engineering can be applied independent of any other services that use labels. Further, the QoS semantics can easily be carried through the stack, allowing consistent QoS to be applied even as traffic engineering is choosing the path that the traffic will take.

Later we'll discuss another use of the label stack to effect fast recovery by allowing a traffic engineered tunnel to be carried within a bypass tunnel.

MPLS Control Components

Many possible control planes can operate in an MPLS environment. These include unicast routing, multicast, Resource Reservation Protocol (RSVP), virtual private networks, frame relay, and traffic engineering.

Multiple control planes can manipulate labels on a single packet; combinations of control planes allow many services. In fact, MPLS could stand for *multipurpose label switching*. Our purpose here is not to explain all of these control components. We merely point out that Traffic Engineering is an important one and can easily be used in combination with others.

We now focus on how traffic engineering employs MPLS.

Applying MPLS to Traffic Engineering

The drawbacks of the overlay model are a result of having distinct layer 2 and networks. The IP network is operating on a virtual topology, which serves to complicate and slow its responses to network events. MPLS allows the elements of traffic engineering to be completely under the control of IP.

We now discuss the various elements of the MPLS traffic engineering solution. This includes the mechanisms for steering packets through the network (label-switched path tunnels), the means of finding appropriate paths through the network (Information Distribution), how traffic is assigned to tunnels, and the ways in which the system responds to topology changes.

LSP Tunnels

The path a packet takes as the result of a series of label operations is called a label-switched path, or LSP for short. LSPs can be established in many ways by various control planes. When we explicitly route an LSP, we call it an *LSP tunnel*. The term derives from the fact that packets travelling along the tunnel have temporarily *tunneled* below the normal mechanisms of IP routing. LSP tunnels share many of the characteristics of ATM VCs. They are explicitly set up and routed. They can have a rich set of QoS mechanisms associated with them.

LSP tunnels are usually established via a signaling protocol such as RSVP. In RSVP, a *path* message carries the explicit route to be followed and is used to temporarily allocate resources along the path. A *reservation* message is sent in response. This establishes the label operations and turns the temporary allocation into a permanent reservation. When using RSVP, the full QoS offerings of integrated services are made available. As the use of RSVP for differentiated services (DiffServ) is defined within the Internet Engineering Task Force (IETF), these will also be available to LSP tunnels. There is ongoing work in the MPLS group to define the use of the three experimental bits in the MPLS header to represent different DiffServ code points. Thus, up to eight different DiffServ codepoints will be available over a single tunnel.

Unlike a VC, LSP tunnels are unidirectional. We refer to the source router as the *head-end* and the destination router as the *tail-end*. This unidirectional nature fits well with engineering IP traffic. In IP, the reverse and forward paths for a flow are independent. Thus, tying the reverse and forward flows in traffic engineering has no value. Further, as a general optimization principle, tying the reverse and forward flows adds unnecessary additional constraints to the problem.

Constraint-Based Routing Information Distribution

In order to establish traffic engineered tunnels in a useful way, they must be routed with sensitivity to the traffic load they will carry. Just as ATM uses the private network node interface (PNNI) to distribute link constraint information, constraint information must be distributed in the IP/MPLS network. Because a consistent database of information is required throughout the network, a means of globally distributing information is required. The flooding used in link-state protocols is one such mechanism.

IP has two link-state routing protocols, Open Shortest Path

First (OSPF) and ISIS. Each of these has been extend to carry link constraint information. Thus, there is no need for an additional layer 2 routing protocol. Constraint and unicast forwarding information are integrated into a single database. IP sees the real topology.

In the overlay model, the virtual links that serve as traffic engineering tunnels are advertised and they create flooding adjacencies over which all link-state updates must flow. This is what creates the n^2 scalability problem. With MPLS, tunnels are not advertised; nor are updates flooded over them.

One further aid to scalability results from this. In the overlay model a single physical link may carry dozens or hundreds of virtual circuits. When such a link fails, the failure appears to IP as the failure of those dozens or hundreds of links. With MPLS, a single link failure looks like a single link failure. Flooding is greatly reduced; convergence can occur much faster.

The Traffic Engineering Control Module

One means of creating a traffic engineering system is a distributed control system. In this scenario, each router has a traffic engineering control module. This application interprets the configured traffic engineering tunnel requirements. Based on those requirements it establishes tunnels. To find a path appropriate for a given tunnel, the module consults the topology database. It then signals for the tunnel to be setup. When the control module is notified that the tunnel is setup, it in turn notifies the routing protocol (ISIS or OSPF) of the tunnel's existence. Thus, the tunnel becomes available for routing traffic.

On a continuous basis, the traffic engineering control module monitors the status of its tunnels. Any tunnel for which setup failed will periodically be retried. The module also looks for indications of any service failures so that corrective action can be taken. To accomplish this it monitors both RSVP error messages and link-state updates.

Link state information is used to accelerate the detection of failures. Hundreds of LSP tunnels may have been traversing a link when it failed. It takes a finite amount of time to create and send those hundreds of error messages. At the same time, the router where the failure occurred will be creating a single link-state update which will rapidly be flooded throughout the network.

Each router's control module keeps a map of which network links are used by which of the tunnels it set up. When a link-state update arrives indicating the failure of a link, the control module can rapidly identify which of its tunnels are affected and begin corrective action. In an overlay environment, this link failure information would only be available to the layer 2 device. This is yet one more benefit of having a single, integrated routing system.

Assigning Traffic to Tunnels

A further benefit of integrated routing is the automatic assignment of traffic to tunnels. To accomplish this a slightly modified Shortest Path First (SPF) calculation is used.

An SPF calculation runs by iteratively placing candidate paths on a "tentative" list, selecting the shortest path from that list, and adding that path (and its destination node) to its forwarding tree. The algorithm begins by adding the root node to the tree and adding the one-hop paths to each of its directly connected neighbors to the tentative list. On each iteration, it adds the current shortest path to its tree and then extends those paths via the links connected to the last node of that path.

To forward IP traffic, a router must determine the interface to the next-hop router. As a router runs an SPF calculation, it caches the interface associated with each path. This begins with the one-hop paths each getting the interface through which they are connected. When a path is extended, the new paths inherit that path's interface.

To automatically route traffic onto tunnels, this algorithm is modified as follows. When the endpoint of a tunnel is reached, the next hop to that node is set to the tunnel interface. As the algorithm proceeds, nodes downstream of the tunnel endpoint inherit that tunnel's interface as their next hop. This will continue until the algorithm encounters another node to which its has a tunnel. The point of this procedure is that traffic is routed loop-free. Only traffic to those nodes which would have been IP routed through the tail-end router is routed through the tunnel. While packets encapsulated on tunnels may flow far and wide in the network, they may only return to the IP level at places where IP routing would have sent them. Thus, the same degree of loop prevention provided by ISIS and OSPF is obtained here.

Other means of assigning traffic to tunnels are both possible and useful. One example is to assign traffic based on BGP next-hop. Another is to assign traffic to different tunnels based on class of service. RSVP defines aggregation over tunnels. LSP tunnels may be used in this way, with the added benefit that they may be routed to where the resources exist if the normal IP route does not have sufficient resources for the request.

Rerouting

Network topologies are never stable over time. A traffic engineering system that expects to deliver good service and remain optimized must respond to changes. Rerouting is necessary over two different timescales. Below we discuss fast rerouting and optimized rerouting. Fast rerouting minimizes service disruptions for the flows affected by an outage. Optimized rerouting serves to reoptimize traffic flows to a changed topology.

Fast Rerouting

Local repair is key to sustaining voice calls and other high-priority services. Transmission and other delays make remote restoration insufficiently responsive. If a repair can be effected local to the device which detects the failure, restoration can be made without serious service disruption. That is, repairs can happen in the same timeframes as synchronous optical network (SONET) restoration.

In MPLS, several techniques are available to enable local repair of LSP tunnels. These are splicing and stacking. In the splicing technique an alternate LSP tunnel is pre-established from the point of protection to the destination via a path that bypasses the downstream network elements being protected. Upon detection of a failure (e.g., a SONET alarm), the forwarding entry for the protected LSP tunnel is updated to use the label and interface of the bypass tunnel.

Another means of local repair takes advantage of the label stack. In this technique a single LSP tunnel is created that bypasses the protected link. One can think of the bypass tunnel as a replacement for the failed link. One this tunnel is established, the local router has a label which represents this bypass tunnel. This tunnel can be used as a *hop* by another tunnel. This is done by pushing the bypass label onto the stack of labels for packet flowing on the rerouted tunnels.

Upon failure of the protected link, all tunnels using that link are updated to use the bypass tunnel. The label forwarding information is updated to first do its normal swap, but then to push on a label for the bypass tunnel and send the packet out the interface for the bypass tunnel. The operation at the next to last hop of the bypass tunnel will be to *pop* the label stack. This will expose the labels expected by the next router for the protected LSP tunnels.

Optimized Rerouting

While fast rerouting is key to nondisruptive service restoral, such restorations often form paths which are not optimal for traffic engineering. Thus, when a failure is detected, it is necessary to also notify the head-end of the LSP tunnel. The head-end can then compute a more optimal path. Traffic can then be diverted to the new LSP tunnel. This can be done without further disruption, as described below.

While responding to the failure of network elements is critical, so is responding to their restoral. An operator wanting to offer the best service possible cannot leave the network in a suboptimal state. Thus, as new (or restored) links appear in the network, we use this as a signal to look for improved routes. Again, if such a route exists, traffic can be nondisruptively redirected to it, as described below.

Rerouting to an Alternate Path

Often missing from layer 2 networks is a feature called *bridge-and-roll* or *make-before-break*. This is the capability to always set up a new VC while maintaining the current VC.

The problem to overcome is this. Suppose the new and existing paths for a tunnel require resources from common links. However, one or more of those links does not have sufficient capacity to admit the second path. Then the tunnel must be first torn down and then reestablished on the new path. If, however, the links were able to recognize the second path as a replacement for the existing path, the path could be admitted.

RSVP has a reservation style called *shared explicit*. This instructs network elements to use the same capacity to service multiple explicitly named sources. In traffic engineering's use of RSVP, we represent a second path for a tunnel as a different *source* by carrying a Path-ID as part of the source identification. When a source (the tunnel's head-end) wishes to reroute, it sends a Path message just as it would for a new tunnel. This message names the same tunnel, but with a new Path-ID. For links not in common, this appears as a new request; but for links that are in common, no new resources need to be allocated.

The tail-end then sends a Reserve message for both paths (senders) using the shared explicit style. The two sender objects are included, and separate label operations are associated with each.

Once the new path is created, traffic is diverted by updating the forwarding table. This occurs without service disruption. The old path can then be removed. The presence of the second path message on shared links prevents the cleanup process from removing resources used by the new path.

Further Optimization Strategies

A number of additional optimization techniques are currently under consideration. These depend on real-time feedback from the network elements. Since the whole system is available to IP, such feedback is much simpler than in the overlay model. In that model, special mechanisms and/or protocols would be needed to communicate between the layer 2 and IP devices.

One optimization strategy is to auto-adjust the bandwidth based on real usage of an LSP tunnel. The head-end router monitors the traffic entering the tunnel. As usage changes, it resizes the tunnel, rerouting it if necessary. Since each router both establishes its own tunnels and maps traffic to them, all the information needed to resize tunnels is readily available.

Another method depends on real-time network link utilization. Each router establishes multiple tunnels to each destination. Based on the utilization of the links in each path, the router load-balances between the alternate paths to even out the traffic load. The necessary link utilization information could easily be added to ISIS and OSPF. In the overlay model, only the layer 2 devices know the link utilization. Further, the particular path of a tunnel is also only known by layer 2. Some means to export this information would need to be devised.

Summary

MPLS has many advantages for traffic engineering. It increases network scalability, simplifies network service integration, offers integrated recovery, and simplifies network management.

Scalability is addressed first by solving the n^2 problem. Only the real topology is advertised. From a routing perspective, the LSP tunnels used for traffic engineering are only known to the head-end router. Further, only one routing protocol is needed; there is no need for a separate layer 2 routing protocol. IP understands the real topology. Assignment of traffic becomes natural; no special mechanisms are required for loop prevention. The churn resulting from a topology change is greatly reduced.

Service integration is simplified by a number of means. First, MPLS offers a single QoS paradigm; but more important, it makes it simple for services to request that QoS and have that request mapped through to traffic engineering. This is achieved through use of the label stack and by mapping the experimental bits down. LSP tunnels can carry multiple kinds of traffic which are differentiated based on the experimental bits. Traffic from multiple control planes can be mapped to a single traffic engineered tunnel. This also enhances the scalability of the system.

Label stacking and tunnel splicing offer effective means for local tunnel repair, enabling fast restoral. The fact that there is one level of control means that recovery can be faster. Feedback through the routing system permits tunnel head-ends to react quickly and intelligently to topology changes. RSVP signaling provides for nondisruptive restoration.

There are fewer boxes and fewer protocols to manage. Configuration and troubleshooting become simpler, reducing overall network management complexity. The need to coordinate and correlate management information from both layer 2 and an IP network is removed.

MPLS offers a more manageable and cost-effective traffic engineering solution. The net result is not only a simpler system; it is a more responsive system, a more scalable system, and a system that can flexibly support multiple service offerings.

Biography

GEORGE SWALLOW (swallow@cisco.com) is co-chair of the IETF Working Group on Multiprotocol over Label Switching. He is also a technical leader at Cisco Systems, where he is a member of the architecture team for Label Switching with responsibility for traffic engineering. Prior to Cisco, he was involved in the design, deployment, and analysis of networks and networking products for BBN, including the Arpanet. He has been participating in the design and standardization of Internet and ATM standards since 1991. He holds an M.S. in mathematics from Northeastern University and a B.A. in mathematics from the University of Virginia.