Computing Packet Loss and Delay using MPLS Trailers

KEVIN MITCHELL
Agilent Laboratories

A technique is presented for accurately measuring packet loss rates for an MPLS label switched path (LSP). The paper describes how loss rates are typically measured in IP flows. It then explores how these techniques can be used in the context of MPLS. The label stack can be used to *tag* particular positions in an LSP flow. The paper then introduces the concept of *packet trailers*, and describes how these could be used as an efficient mechanism for collecting performance readings as a packet traverses an LSP.

Key Words and Phrases: MPLS, active measurement, delay, loss

1. INTRODUCTION

Consider the problem of accurately measuring packet loss on a flow between two points in a network. In some cases this problem is quite easy. In the case of a microflow, where all packets are of the same type, the protocol in use may help. For example, a TCP flow will contain sequence numbers that can assist in detecting packet loss. But even in this case we must be careful not to confuse reordered packets with lost packets, particularly when the measurement points are not co-located with the originator and destination of the flow.

For more aggregated flows, or where the protocol is less helpful, the situation is often more complex. Routers can sometimes be configured to capture packet and byte counts at the flow level (e.g. NetFlow[Cisco 2002]). But whilst such information can be used to estimate traffic rates, it is less useful for determining loss rates in mid-flow due to the difficulty of sampling the counters at both monitoring points as the same packet passes by. Furthermore, the information obtained from MIBs is often slightly stale in some router architectures, making an approach based on SNMP even less attractive. In the absence of signalling, the point at which a router starts to monitor a flow may also vary, further complicating the analysis of these readings. Measurements based on this technique will therefore tend to give only course-grained loss rates that tell us little of how the rate varies over finer time intervals.

Other approaches rely on using hardware probes and hashing techniques. In its simplest form the probe computes a hash for every packet that passes by. Both ingress and egress probes use the same hashing function, and agree on a particular hash value N. Every time a packet hashes to N the probe records the current packet count total for this flow. If the hash function is discriminating enough

Author's address: K. Mitchell, Agilent Labs Scotland, South Queensferry, Scotland EH30 9TG

this technique can allow these counts to be correlated between the two probes to compute accurate loss rates. The packets that pass the test mark points in the stream and the probes can then synchronise their measurements at these points. The sophistication required in such techniques arises from the need to control the rate at which packets can pass the test when the monitoring points have no control over the makeup of the packets in the flow. If the matched packets are too close together then the readings can become ambiguous, particularly when packets are frequently lost. If a packet rarely passes the test then our ability to measure loss rates on a fine timescale is reduced.

In some cases we can simplify the implementation by injecting test packets that can be easily and uniquely recognised by the probes. This avoids relying on adaptive hashing techniques as the rate at which these packets are generated is now under our control. Of course we have to ensure that these packets do not disrupt the recipient(s) of this flow of packets. For a single microflow this may be impossible. But for an aggregated flow, with many recipients for the individual microflows, adding an addition microflow of test packets should cause no disruption. However, without reconfiguring the routers, it may be difficult to guarantee that the injected packets will be treated identically to all the other packets in the flow. They may follow a different route to the destination, or be subjected to different queuing treatment, for example. This increases the likelihood of packet reordering, complicating the loss analysis. Some approaches simplify the solution even further, trying to estimate the overall loss rate by the loss rate experienced by the artificially injected packets. Apart from the need to inject high volumes of active traffic to achieve statistically reliable results, the potentially different QoS treatment of such packets makes it hard to draw any firm conclusions from such results.

A hybrid approach is also possible when a monitoring point is co-located with the source of a microflow. Special IPv4 options or IPv6 header extensions can be used to tag a user packet for monitoring purposes without disrupting the end-point of the flow, as long as the MTU is not exceeded[Jeong et al. 2002]. The hashing function can then be designed to recognise the presence of such headers, or the destination host can be configured to extract and process these headers. They can also be generated at a rate under the control of the management process. A kernel module could be used to introduce such options/extensions in the source machine. Such a technique is more problematic where the monitoring point is downstream of the point where packets are generated. A passive probe clearly can't modify the packets as they pass by. Furthermore adding extension headers to user packets as they pass through a router may have some undersirable ramifications.

2. CALCULATING LOSS RATES FOR MPLS PATHS

From the previous discussion we can see that calculating accurate loss rates for IP flows is not as simple as it might at first seem. We now consider the situation where the flow consists of aggregated microflows transported across a core network using an MPLS label switched path (LSP). We would like to accurately measure loss rates across this LSP. A good example of such a requirement is where these paths are being used to support a VPN with a service level agreement specifying acceptable loss rates. An additional requirement is that any solution should have

minimal impact on the system under test, i.e. it should have negligible impact on the packet forwarding rate of the router elements.

At the ingress of an MPLS label switched path a packet is encapsulated inside an MPLS shim header. This header is placed between the layer two header, e.g. an ethernet header, and the layer three header, e.g. an IPv4 header. The shim header, illustrated in Figure 1, contains a stack of entries. Each entry contains a label, an EXP field and a TTL field. One bit, the S field, is used to indicate the bottom of the stack. Initially, when an IP packet enters an MPLS cloud, a single shim entry

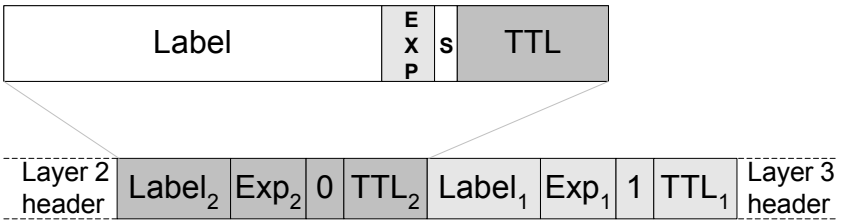

Fig. 1. The MPLS shim header

is added to the packet. However, whenever an MPLS packet is tunneled through a link implemented by another LSP the label stack is extended. When the egress of the tunnel is reached the top-most entry is popped. At the egress of the MPLS cloud the final entry is popped and the packet continues on its way using level three routing.[1]

The treatment of an MPLS packet at each router depends solely on the values of the label and EXP fields of the top-most entry in the shim header. The label determines the outgoing interface(s) and label(s) to use, i.e. the routing of the packet. The EXP field can be used to support multiple qualities of service across this path, by influencing queuing behaviour for example. This makes it easy to introduce additional monitoring packets, safe in the knowledge that as long as they have the same top-level shim entry as packets in the flow of interest then they will be treated identically to all other packets in this LSP. Furthermore, when these packets reach the egress of the LSP the uncovered IP header can direct these packets to a monitoring station without disturbing the "user component" of the flow.

As in the case of IP measurements, we can inject monitoring packets into the LSP, marking specific points in the flow, and then generate statistics whenever these packets pass a monitoring point. How can we mark these packets so they can be recognised efficiently? Whilst the level three and four headers may contain this information, performing such deep matching in a core MPLS router, or external probe, is likely to be expensive and/or time-consuming. Fortunately we can exploit the shim header label stack for tagging purposes. The label field is twenty bits long, and most labels are assigned via label distribution protocols, or manually. However, the first sixteen values are reserved for special uses[Rosen 1999]. A value of 0 represents the "IPv4 Explicit NULL Label". This value indicates that the

[1]Penultimate hop-popping can complicate this simple description slightly.

label stack must be popped, and the forwarding of the packet must then be based on the IPv4 header. A value of 2 has a similar semantics, but for IPv6 packets. These labels only make sense when they are the bottom labels in the stack. Strictly speaking, they are also only legal when they are the sole label stack entry. However, for now let us relax this rule, allowing other labels to appear on top of them in the label stack.

Suppose we inject an MPLS packet into the ingress of an LSP, with an "Explicit NULL" label *underneath* the ingress label for this LSP. We call such a packet a *monitoring packet*. We assume the existence of a mechanism for injecting such packets into an LSP, either through label merging, tunneling, local modification of the ingress forwarding table, or a more specialised technique introduced for monitoring purposes. Such a packet would traverse the LSP until it reached the egress. The presence of the extra label would have no influence on the treatment of the packet at each hop. At the egress the top shim header entry would be popped, exposing the Explicit NULL label. This would also be popped and the packet forwarded using the level three header. An external monitor, or the routers themselves, could easily detect the presence of the additional shim entry, as it would always be at a fixed offset from the top of the stack, and have a fixed value. This makes it an ideal tag for a monitoring packet. If the LSP is tunneled over another LSP at some point along the path, then the Explicit NULL entry will move further down inside the stack, and so the packet will not be recognised as having a tag. However, this is the behaviour we want as a transit provider, supplying an LSP for tunneling purposes, does not want the structure of their tunnels exposed to external monitoring applications. We will return to this point later.

The basic idea of using the MPLS label stack to tag monitoring packets could be instantiated in a number of different ways, depending on the hardware being used. We now present some of these alternatives, including a new one based on *packet trailers*.

2.1 External probes

The simplest approach, relying on no support from the routers themselves, uses external probes. The probes would passively monitor the flow of packets on a link and maintain per-LSP packet counts. Whenever a monitoring packet was recognised the current packet count for the LSP would be sent to a correlator. Two probes, one on the outgoing link of the ingress LSR, and the other on the incoming link of an egress LSR, could be used to calculate accurate loss rates across the LSP. If the ingress probe was also responsible for generating the monitoring packets then these packets could be used to transport previously observed readings from the probe. The egress probe can then calculate the loss rates without needing a separate correlator.

As mentioned earlier, strictly speaking labels 0 and 2 should only be used when they are the only entry on the stack. Routers between the ingress and egress will only examine the top label on the stack, and so this violation will not be noticed. However, when the egress router pops the top label, and then finds the label 0 or 2 on the top of the stack, the router may reject such a packet. It will obviously have code to handle these labels when received on their own, so whether a router will be pedantic and reject the packet, or handle the packet in the obvious way,

will clearly be router/vendor dependent. If a router was found to treat these labels strictly then other labels, and tunneling, could be used to achieve a similar result. This would complicated the probes slightly though, as it would no longer be able to match on a fixed label in the stack.

An external probe has a number of obvious deficiencies. Probes may be quite expensive, particularly when monitoring high-speed links in the core of a network. There will usually be far fewer probes deployed than links. Furthermore, as LSPs adapt to link and router failures, the exact path used by an MPLS tunnel may vary over time. This may cause a probe to "lose" an LSP if the probes are not deployed on sufficient links. A more cost-effective, and pervasive, solution can be obtained by moving the monitoring functionality into the routers themselves. In doing so, it is critical that the forwarding speed of the devices is not adversely affected. There are a number of approaches that could be used.

2.2 A MIB-based solution

An MPLS router will need to keep packet and byte counts for each LSP in order to support the LSR MIB[Srinivasan et al. 2001]. Furthermore, separate counts are maintained for the in-segment and the out-segment. When a packet arrives at a router interface it is added to the in-segment total. When/if it leaves the router, typically via another interface card, it is added to the out-segment total. We introduce a new reserved label, 4 say, with the same semantics as label 0, but without the restriction that it must be the only label on the stack. Detecting monitoring packets using this label, in the manner described earlier, is clearly very simple, whether done in hardware, firmware, or software. Whenever such a packet passes a point in the router where one of the packet counts is incremented we cache the counter's value. To retrieve such cached values we can use a MIB designed for this purpose, e.g. via augmentations to the InSegment and OutSegment performance tables. Each monitoring packet injected into an LSP will result in the cached values of the counters being overwritten. However the generation of the monitoring packets is under the control of the management station. A new monitoring packet for this LSP will only be generated after the counters have all been read from the MIBs.

This technique could also be used to monitor loss rates across the entire LSP, not just at the ingress or egress. However, this would require more administrative work to keep track of where the LSP is currently routed, and the labels being used for each segment. The amount of MIB traffic is also increased. A better solution, and the main topic of this paper, uses the monitoring packets themselves to carry this information.

2.3 Packet trailers

Whilst traversing an LSP the details of any encapsulated IP header are treated opaquely. In particular, the IP header length field is not used, and the length of the packet is determined, either explicitly or implicitly, by the link layer protocol at each hop. We exploit this behaviour in the following way. At each point where an MPLS monitoring packet is recognised some additional data is appended to the end of the packet. This data would include at least the router interface address, packet count and timestamp. We call this additional measurement data a *trailer*.

This process is illustrated in Figure 2.

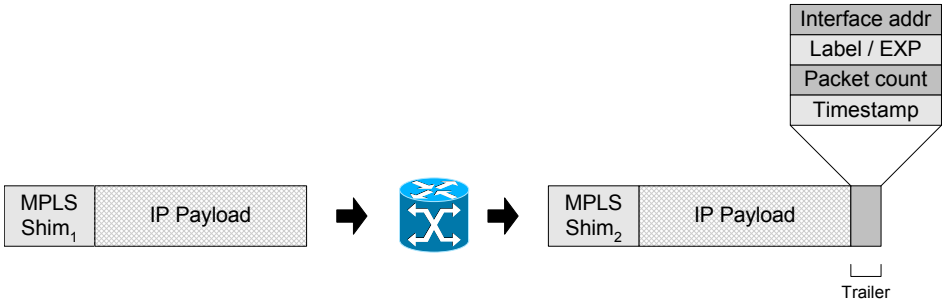

Fig. 2. Adding a trailer

The payload encapsulated by the MPLS header is not altered in any way by this process until the packet reaches the egress router. When the top label is popped, exposing the monitoring label, the shim header is removed. The IPv4/v6 header is then updated to merge the additional trailers into the packet payload. The IP packet length field is modified, as is the header checksum. If we make the simplifying assumption that the packet is carrying UDP data, which is a reasonable assumption as we control the generation of the packets, then the UDP checksum, if any, is also checked and recomputed. This process is illustrated in Figure 3.

Fig. 3. Egress merging trailers into the payload

What does such a process achieve? First note that the treatment of any MPLS packets that don't use this special label is unchanged. A router implementing this proposal would therefore have no effect on user data. Imagine injecting a monitoring packet into the LSP with an empty UDP payload. On exit from the LSP the packet would have been transformed, and would now contain trailers with measurements for each hop along the path. There are typically two points in a label switching router (LSR) where counters are updated for a packet, illustrated in Figure 4.

In the best case, where every router along the path supports this technique, there will be one trailer generated for the ingress router, two for each intermediate router, and one for the egress router. Obviously routers that were unaware of such monitoring labels will not add trailer entries, and so some hops will be invisible. Similarly LSP tunneling would appear as a single hop. This protects the privacy of any transit LSPs, an important feature for service providers. A monitoring packet grows as it passes through an LSP. Although unlikely, an LSP with a large number

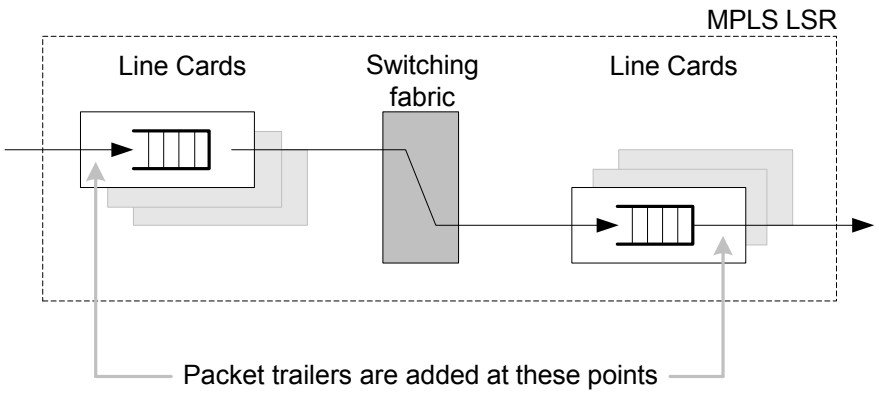

Fig. 4. Trailer creation points

of hops could result in the packet length exceeding the MTU of one of the links. The shim header TTL field can be used to limit the number of hops traversed by such packets. Given the MTU for the LSP, and the size of each trailer, it would be simple to compute a suitable value for the TTL field when the monitoring packet is generated.

If the egress router doesn't support the new "monitoring" label the packet will presumably be discarded. We can use this to test whether the router supports the technique. Note that the cost of supporting trailers at each hop is very minimal. We just append data that is already available to the end of the packet. For some link layer technologies we can make the decision on whether this is a monitoring packet in parallel with sending the start of the packet. This proposal should therefore have a negligible effect on the packet forwarding rate of each LSR. It is only at the egress where we have to spend a bit more time incorporating the extra data into the packet, fixing up the IPv4/6/UDP header. After that the packet can be routed to a separate management station at leasure as the time/position critical data has been stored in the packet.

The packet trailers are not protected by the IP header checksum, or the UDP checksum if present. We could add checksums to each trailer, and optionally check and remove these when the trailers are consolidated into the payload at the egress. Alternatively it may be simpler to just rely on coping with a small number of erroneous results in the management station, particularly as the errors will not be cumulative.

3. SMART GBICS

Ideally you would like to detect the measurement packets as soon as they arrive at a router, before any queuing takes place. Similarly when leaving a router you would like to update these packets just before they were placed on the wire. By doing this you get an accurate measurement of the cross-router delay. A recent proposal for performing packet measurements on Gigabit networks places some packet filters and counters in the Gigabit Interface Converter (GBIC), a hot-swappable input/output

device that plugs into a Gigabit Ethernet port or slot, linking the port with the network. This is an ideal place to time the arrival and departure of packets, as we have pushed the measurement point to the very edge of the router. The simplicity of the packet trailer approach makes it an ideal candidate for one of the tasks performed by a "smart" GBIC. However, it does create some problems. Whilst it can alter the bits in a packet as it passes by, and conceivably truncate a packet, it can't increase its length. Without a feedback mechanism to pause the devices driving the GBIC, we could only add additional bits to a packet by introducing buffering. We wish to avoid this as it introduces additional delay for all packets, and because there is no natural bound to the size of buffer required. For these reasons the packet trailer idea requires some modification in the context of these devices. The obvious approach is to pad out the monitoring packet at the time of injection with sufficient room to hold all the trailers. An MTU-length packet would suffice here. We would then use a "free pointer" at a fixed offset, to record where to store the next trailer. The pointer would be updated each time a new trailer was added to the packet. This makes the process of updating the packet with each trailer slightly more complex, but the GBIC incarnation of the approach has the advantage of being able to upgrade existing routers with this measurement capability relatively cheaply.

4. CONCLUSIONS

The main proposal of this paper is that with a small amount of additional hardware, or firmware, an MPLS router could support a powerful performance monitoring facility without any detrimental effect on forwarding performance. For some link layer technologies it may be possible to support such a facility in a separate optional chip, particularly if the per-LSP packet counting is duplicated, thus greatly simplifying the interfacing issues. In the case of "smart GBICs" we may even be able to add such functionality to an existing router.

The ability to efficiently measure loss and delay across an LSP, not just at the end-points, would give us a powerful diagnostic tool. By only detecting the monitoring label when it is the second label in the stack we protect the confidentiality of any transit providers. Although the primary interest is in calculating packet loss, we can also generate delay measurements. To calculate delay between two routers would require their timers to be synchronised in some fashion, e.g. [Horauer and Höller 2002]. Measuring the internal delay, i.e. the time between the generation of the in-segment trailer and the out-segment trailer, obviously requires no such synchronisation. In core networks, with routers connected by point-to-point links, almost all the variability in end-to-end delay will be due to the variable delays within the routers due to the changing network loads, and their impact on the router queues. By using fixed delay estimates for the links, and accurate cross-router delays derived from the packet trailers, we may be able to construct reasonably accurate estimates of end-to-end delay in such cases without requiring the complexities involved in distributed time synchronisation.

The approach is more efficient than using MIBs to collect such information. A single monitoring packet can collect information from all routers along the path, and the results are automatically forwarded to the management station. In a QoS-

aware network one monitoring packet would need to be injected for each of the different settings of the EXP bits currently in use.

All information is valuable, and so there are obvious security implications with such an approach. Only trusted management stations should be allowed to commission such measurements. The approach relies on the generation of a monitoring packet, with a reserved label inside the shim header. We have not addressed how such packets can be generated. There are a number of possible mechanisms that could be used. Manually configuring the ingress tables, via SNMP, is one such approach. Any user with sufficient authorization to make such changes is presumably trusted enough to make these measurements. Thus the security of this approach can be delegated to the security of the mechanisms necessary to generate the monitoring packets themselves.

The trailer technique could also be applicable in a non-MPLS setting, although not as elegantly. For example, at what point along the path should the additional data be consolidated into the packet body when there is no explicit egress node? The cost of recognizing monitoring packets, e.g. using IPv4 options, or IPv6 header extensions, together with the cost of consolidating trailers at each hop in the worst case, makes the technique much more expensive in the IP setting.

Techniques such as network tomography attempt to calculate loss rates for internal nodes of an IP or MPLS cloud from end-to-end measurements across the cloud. Extensive deployment of the trailer technique could avoid the need for such approaches, and could produce much more accurate results, particularly in a QoS-aware network.

REFERENCES

Cisco. 2002. Netflow services and applications white paper. www.cisco.com.

Horauer, M. and Höller, R. 2002. Integration of high accurate clock synchronization into ethernet-based distributed systems.

Jeong, J., Park, J., Lee, S., and Kim, Y. 2002. One-way delay measurement using IPv6 source routing. www.ietf.org/internet-drafts/draft-jeong-1way-delay-ipv6-source-routing-00.txt.

Rosen, E. 1999. MPLS label stack encoding. www.ietf.org/internet-drafts/draft-ietf-mpls-label-encaps-07.txt.

Srinivasan, C., Viswanathan, A., and Nadeau, T. 2001. MPLS label switching router management information base using SMIv2. www.ietf.org/internet-drafts/draft-ietf-mpls-lsr-mib-07.txt.