May 2023

APPLICATION-DRIVEN OPTIMIZED SLA-AWARE PATH SELECTION FOR COLLABORATION APPLICATIONS

Dave Zacks

Paul B Giralt

Prapanch Ramamoorthy

Thomas Szigeti

APPLICATION-DRIVEN OPTIMIZED SLA-AWARE PATH SELECTION FOR COLLABORATION APPLICATIONS

AUTHORS:
Dave Zacks
Paul B Giralt
Prapanch Ramamoorthy
Thomas Szigeti

ABSTRACT

Currently, applications are at the mercy of a network's infrastructure for the selection of a path within a network environment where more than one path exists between a source and a destination. Too often, the network infrastructure elements are unaware of an application's requirements, or are aware of them in only a very rudimentary way. This situation is particularly dire for collaboration applications, which often have the most stringent requirements for path characteristics including delay, jitter, and packet loss. Techniques are presented herein that move the point of control for path selection to a collaboration application through a lightweight, in-band signaling mechanism that is exposed by the application to a network's infrastructure for appropriated and differentiated traffic routing. Aspects of the presented techniques support the use of a per-application tunneled path for traffic flows, combined with a measurement methodology for those multiple paths and a mechanism for the application-level designation of specific and differentiated traffic pathing via an upstream router, allowing an application to measure performance across multiple paths and then signal to a network which path to choose based on per-application preference and service-level agreement (SLA) criteria.

DETAILED DESCRIPTION

Currently, applications are at the mercy of a network's infrastructure for the selection of a path within a network environment where more than one path exists between a source and a destination (which includes, of course, almost all of the networks in the real world). Too often, those network infrastructure elements are unaware of an application's requirements, or are aware of them in only a very rudimentary way (through, for example, an examination of differentiated services code point (DSCP) value settings, which lack per-

application granularity). As a result, applications are often shunted over only a single path within a network (typically the "shortest path" according to routing metrics) even when a better path (such as, for example, a path that is better suited to the actual application requirements) exists.

This situation is particularly dire for collaboration applications, which often have the most stringent requirements for path characteristics including delay, jitter, and packet loss. Such collaboration applications may end up with their traffic sent over a less optimal path, even when a more optimal path exists.

Presented herein are techniques to address this type of challenge that may operate in a lightweight fashion and impose no undue burden on either an application or the supporting network infrastructure. In particular, the presented techniques move the point of control for path selection to a collaboration application through a lightweight, in-band signaling mechanism that is exposed by the application to a network’s infrastructure for appropriated and differentiated traffic routing.

In an end-to-end path between a collaboration client and an associated service or server to which that application is communicating, no point is better suited to understanding the needs of the application than the client itself. Video endpoint applications know what their needs are for a network path in terms of latency, jitter, and loss boundaries, as do whiteboard, telephony, and other collaboration applications. The need for optimized, bounded conditions for network path selection becomes even more important, and more stringent, as augmented reality (AR), virtual reality (VR), and similar applications become extant within networks over the next several years.

In connection with the techniques presented herein, it is important to examine and understand the nature of traffic flows in today's Internet. Currently, most networks (corporate as well as personal) involve one or more network address translation (NAT) boundaries that connect private to public address spaces. When a (first-hop or midspan) router has multiple NAT'd uplinks, any set of given flows is bound to that uplink in terms of being NAT'd into that uplink's address space. In the event that all or a subset of those flows are switched to another NAT'd uplink, the involved connections break since the Internet Protocol (IP) addresses necessarily change to the new NAT'd address space. This is one important reason why performing traffic rerouting today at such a boundary is

impractical and disruptive. Yet, this boundary point is often exactly where the choice between two service-level agreement (SLA)-varying paths must be made.

Consider the not-uncommon scenario where a business, or even an individual user, has two Internet uplinks (which may be referred to as Uplink A and Uplink B) from two different providers, as illustrated in Figure 1, below.

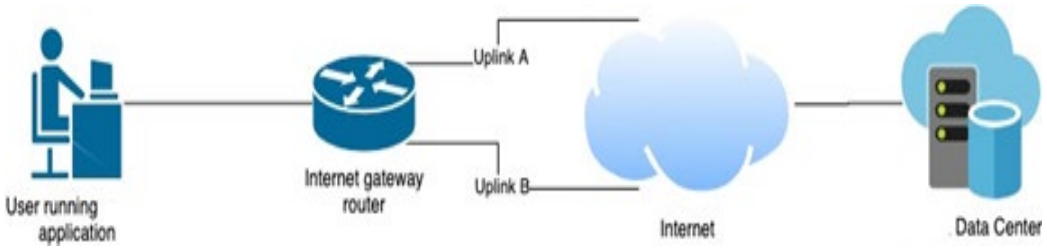

Figure 1: User with Multiple Uplinks to Internet and Cloud(s)

Such an arrangement is often pursued today for better resiliency, as most organizations and users today "live and die" by their Internet connectivity, especially since the bulk of today's applications are deployed in the cloud. Having more than one network connection is even fairly common in consumer networks. For example, a mobile device may have both a Wi-Fi and a cellular connection, or a home router may have two Internet service provider (ISP) connections such as a broadband and a cellular or low Earth orbit (LEO) satellite connection.

Referring again to Figure 1, above, in the event of a decision to reroute some or all of the traffic from Uplink A to Uplink B the NAT'd IP addresses of that traffic will necessarily change, breaking the end-to-end Transmission Control Protocol (TCP)-based connections and the bidirectional media flows that are reliant on return traffic being directed to the public NAT IP address, thereby causing disruption, disconnections, etc.

While some collaboration applications have been designed to deal with such IP address changes, this necessarily involves a break in the traffic flow, typically amounting to several seconds. Indeed, for many other collaboration (and additional) applications, this irreparably breaks the end-to-end connection, thus necessitating session reestablishment after the break in the connection. In either case, this makes traffic rerouting impractical at

this boundary unless in the event of an entire uplink failure (which, as a consequence, is typically all that such redundant pathing is used for today).

Under aspects of the techniques presented herein, a collaboration application may tunnel its traffic in a lightweight client-server fashion. Such a tunnel may take multiple forms (such as a Virtual Extensible local area network (VXLAN), a Generic Protocol Extension for VXLAN (VXLAN-GPE), Generic Routing Encapsulation (GRE), etc.) with the only caveat being that the tunnel header must have at least one exposed bit that may be used for path selection (as described in detail below). Since the tunnel header is by definition unencrypted, that bit will always be readable and interpretable by network devices in the end-to-end path.

According to the techniques presented herein, there are a number of candidate bits that may be used in a tunnel header to indicate a path selection (between, for example, a first path and a second path).

A first candidate is DSCP bit six. All of the Internet Engineering Task Force (IETF) standard per-hop behaviors are drawn from Pool 1, as defined in section six of the IETF Request for Comments (RFC) 2474 (i.e., the final bit of these codepoints is always set to zero (0)). This leaves Pools 2 and 3 available for experimental or local use, which means that this final bit in the DSCP field may be utilized for path selection. For example, if the bit is set to 0 then a first path may be selected and if the bit is set to one (1) then a second path may be selected.

A second candidate involves the IP type of service (ToS) bits six and seven. These bits have been designated to signal IP Explicit Congestion Notification (ECN), as defined in the IETF RFC 3168. The first of these bits indicates an ECN-Capable transport (i.e., the network device supports IP ECN) and the second bit indicates Congestion Experienced (CE). This leaves a bit combination that is logically contradictory – specifically, the 01 bit combination which would be interpreted as a device that is not ECN-Capable but which has experience congestion and is using the ECN fields to signal the same. As such, network devices could be programmed to use this special bit combination to signal the selection of a second path, with any other ECN combination used to signal the selection of a first path.

A third candidate encompasses VXLAN and other tunneling protocols which support flexible fields that may be used to convey path selection preferences.

Figure 2, below, presents elements of a tunneled path connection according to the techniques presented herein and reflective of the above discussion.

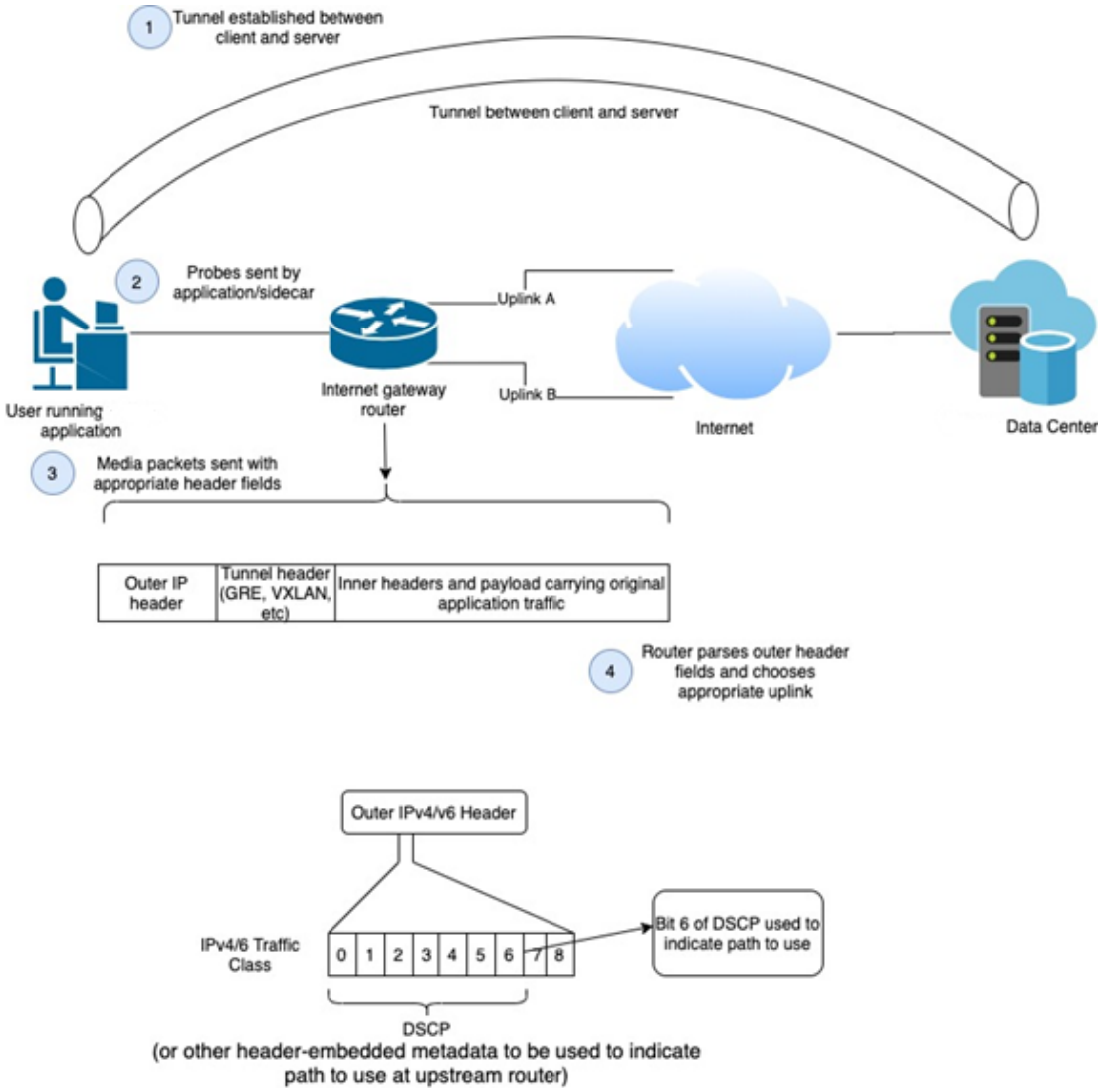

Figure 2: Operation of Tunneled Path Connection Combined with Path Selection Metadata for Upstream Router Use

The approach that was described and illustrated above alleviates the issue of IP address changes in the event of a traffic rerouting decision at any NAT boundary. Since the receiving server will pop-off and discard the tunnel header at the destination, and the inner IP address of the client will remain unchanged, the rerouting of traffic to a different routed uplink will not affect the end-to-end traffic flow.

According to the techniques presented herein, the use of a tunnel header also allows for the insertion of one or more path selection bits by a collaboration application. This allows the application to signal to an upstream device (such as, for example, the application's first-hop router) as to whether the traffic from this specific collaboration application and a traffic flow should follow a first path or a second path upstream.

An upstream router may then perform a very basic and simple path selection decision. Based on the indicated bit setting, a packet may be sent down a first link or a second link. No deep packet inspection (DPI) or other central processing unit (CPU)-consuming process is required at that router.

If only a first link is available (in a case where, for example, a second link is down), traffic may be sent over the available link regardless of an application tunnel setting. However, in the event that multiple paths are available the application is now in direct control of which path is selected.

It is important to note that it is not necessary for an application to be informed by a router as to which paths may be available or how many such paths exist. The techniques presented herein will operate without any such formal interaction between a first-hop router and a client, based simply on the client assuming that multiple paths may be available and marking the tunneled traffic appropriately. However, if such an interaction were put in place between the router and the client (for example, the router informing the client as to how many such paths may exist) then the operation of the path selection method at the application-level could be further enhanced.

Additionally, the techniques presented herein do not require that a router measure the performance of the available paths. Such measurements may all be done at the application level, or possibly by using a "sidecar" or adjunct application at a client device as noted below. By so doing, the processing load on the network device is kept to a minimum. This is an important consideration with today's high-speed network devices, which often operate with relatively-inflexible hardware-based data plane implementations or with limited available CPU horsepower.

An application may determine which path it wishes to use for its traffic flows by periodically generating probe packets, marked with a tunnel header setting indicating the use of a first path or a second path, that are directed to its designated destination service

and then measuring the responsiveness (such as latency, jitter, loss, etc.) over each path of the probe responses by that service. By employing such an approach, the application itself (based on its own criteria) can decide which path best meets its needs and appropriately mark the application's actual tunneled packet headers.

In the event that an involved router cannot correctly interpret tunnel header bit markings, it may simply forward the traffic in an unaltered fashion (typically, over the default path according to its routing table). This will result in an operation that is no worse than it is today in terms of quality of service (QoS). However, as mentioned previously, such an approach will still optimize failover in terms of eliminating switchover traffic disruptions for collaboration applications that are augmented with the techniques presented herein (and thus make use of a tunneled path).

An application may drive its own path performance measurement capabilities, as noted above, or it may instead leverage an enhanced version of a path measurement capability in a network performance facility that has implemented the techniques presented herein. In such a case, a "sidecar" application may be set up to enable simulated tunneled collaboration application traffic, measuring the various available paths and then providing, through an application programming interface (API), the collected empirical metrics to any collaboration applications that are running on the involved device. Those applications, thus alleviated of the measurement function, can therefore simply mark their tunneled traffic in a lightweight fashion and reap the benefits of the techniques presented herein for optimized path selection for their traffic flows on an ongoing and continuously updated basis to meet their SLA requirements.

For applications that require the highest levels of reliability, an application may simultaneously send and receive media streams across both paths by simultaneously sending packets over both connections. The receivers on both ends would be responsible for the de-duplication of packets, functionality that already happens inherently with sequence numbers in the Real-time Transport Protocol (RTP) for real-time media as well as in a TCP stream.

Even for applications that do not require sending duplicate media streams over both paths simultaneously, they can still establish signaling connections over both channels and open any necessary media ports through a NAT operation by utilizing the Interactive

Connectivity Establishment (ICE) and Session Traversal Utilities for NAT (STUN) protocols at the initiation of a media session. Under such an approach an application can quickly fail over to the second connection, in the event of detecting a failure, without having to go through the motions of setting up the new signaling connection or the media path of the second network path, thereby speeding up any failover. A client may send periodic packets on the negotiated signaling or media ports to ensure that firewalls do not time-out the connections, which can be common in the case of the User Datagram Protocol (UDP).

If the ICE protocol is employed for media negotiation, a client may send STUN packets out of both paths and possibly receive server reflexive candidates from both NAT addresses. These could both be advertised as valid media paths for connectivity checks.

According to aspects of the techniques presented herein, no tunnel header is needed at all. Instead, DSCP or IP ToS bits in the native media or signaling packets that are generated by a client may be tagged (as described previously) and used for the sole purpose of path selection by the upstream routing or NAT device. If NAT is being performed at such a device, the return traffic will automatically come back on the correct interface because it will be directed to the NAT address of one of the two interfaces.

It is important to note that while primarily focused on collaboration and real-time media use cases, the techniques presented herein are applicable to any application that wishes to have some level of control over redundant network connections in an environment similar to what has been described above.

According to further aspects of the techniques presented herein, a shim, proxy, or agent that is running on a client could perform the above-described path selection actions on behalf of a single application or all of the applications that are running on the client. For example, a virtual private network (VPN) client may simultaneously establish connections to multiple destinations over both paths by signaling which path to use for each of the connections and then (completely transparent to the applications that are running on the client computer) direct traffic to one path or the other path on behalf of applications based on different criteria (as explained above). The tunnel connections may terminate at two different cloud locations and provide high availability to all of the applications running on the computer.

As described and illustrated in the above narrative, the techniques presented herein support an automated, lightweight, and simple method that provides collaboration applications with a number of capabilities. First, an application may sense the end-to-end performance of its own traffic flows in the multi-path, NAT'd network environments that are common today. Second, an application may select a given path that meets its SLA requirements based on empirical metrics. Third, an application may signal in a lightweight fashion to an upstream router which path to select for any given set of packets or flows. Importantly, each of these capabilities are optimized for minimal to no disruption of traffic in the event that traffic is rerouted.

While the techniques presented herein may be applied to any application, they are particularly critical for collaboration applications, which are currently heavily leveraged by all organizations. The use of a per-application tunneled path for traffic flows, combined with a measurement methodology for multiple paths and a method for the application-level designation of specific and differentiated traffic pathing through an upstream router, are central to the presented techniques and combine to achieve a capability that is not possible today.

The techniques presented herein would be extremely useful to a network equipment vendor and its customers, as it would allow for the optimized performance of vendor-supplied or third-party applications over an application-aware network infrastructure, enhancing the connection between, and the value proposition of, an intelligent, application-aware network infrastructure, thus helping to drive a vendor’s core value proposition around secure, intelligent networks.

In summary, techniques have been presented herein that move the point of control for path selection to a collaboration application through a lightweight, in-band signaling mechanism that is exposed by the application to a network’s infrastructure for appropriated and differentiated traffic routing. Aspects of the presented techniques support the use of a per-application tunneled path for traffic flows, combined with a measurement methodology for those multiple paths and a mechanism for the application-level designation of specific and differentiated traffic pathing via an upstream router, allowing an application to measure performance across multiple paths and then signal to a network which path to choose based on per-application preference and SLA criteria.