Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 19, 2016

M. Kuehlewind, Ed.
B. Trammell, Ed.
ETH Zurich

March 18, 2016

Use Cases for a Substrate Protocol for User Datagrams (SPUD)
draft-kuehlewind-spud-use-cases-01

Abstract

This document identifies use cases for explicit cooperation between endpoints and middleboxes in the Internet under endpoint control. These use cases range from relatively low level applications (improving the ability for UDP-based protocols to traverse firewalls) through support for new transport services (in-flow prioritization for graceful in-network degradation of media streams). They are intended to provide background for deriving the requirements for a Substrate Protocol for User Datagrams (SPUD), as discussed at the IAB Stack Evolution in a Middlebox Internet (SEMI) workshop in January 2015 and the SPUD BoF session at IETF 92 in March 2015.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describe use cases for a common Substrate Protocol for User Datagrams (SPUD) that could be used by superstrate transport or application protocols to explicitly expose information to and exchange information with middleboxes about application traffic and network conditions.

For each use case, we first describe a problem that is difficult or impossible to solve with presently deployable protocols within the present Internet architecture. We then discuss which information is exposed by endpoints about the traffic sent, and/or by SPUD-aware middleboxes and routers about the path that traffic will traverse. We also suggest potential mechanisms to use that exposed information at middleboxes and/or endpoints, in order to demonstrate the feasibility of using the exposed information to the given use case.The described mechanisms are not necessarily proposals for moving forward, nor do they necessarily represent the best approach for applying the exposed information, but should illustrate and motivate the applicability of the exposed information. We further discuss incentives for deployment and any security, privacy, and trust issues that arise in exposing and/or making use of the information.

1.1. Principles and Assumptions

We make a few assumptions about first principles in elaborating these use cases

1.1.1. Trust and Integrity

In this document, we assume no pre-existing trust relationship between the communication endpoints and any middlebox or router on the path. We must therefore always assume that information that is exposed can be incorrect, and/or that the information will be ignored.

This implies that while endpoints can verify the integrity of information exposed by remote endpoints, they cannot verify the integrity of information exposed by middleboxes. Middleboxes cannot verify the integrity of any information at all. In limited situations where a trust relationship can be established, e.g., between a managed end-user device in an enterprise network and a corporate firewall, this verifiability can be improved.

1.1.2. Endpoint Control

We further assume that all information exposure by middleboxes happens under explicit endpoint control. For that reason, the information exposed by middleboxes in this document takes only two forms. In the first form, "accumulation", the endpoint creates space in the header for middleboxes to use to signal to the remote endpoint, which then sends the information back to the originating endpoint via a feedback channel. In the second form, the middlebox sends a packed directly back to the endpoint with additional information about why a packet was dropped. Other communications patterns may be possible, depending on the first principles chosen; this is a subject of future work.

1.1.3. Least Exposure

Additionally, this document follows the principle of least exposure: in each use case, we attempt to define the minimum amount of information exposed by endpoints and middleboxes required by the proposed mechanism to solve the identified problem. In addition to being good engineering practice, this approach reduces the risk to privacy through inadvertent irrelevant metadata exposure, reduces the amount of information available for application fingerprinting, and reduces the risk that exposed information could otherwise be used for unintended purposes.

2. Firewall Traversal for UDP-Encapsulated Traffic

We presume, following an analysis of requirements in [I-D.trammell-spud-req], as well as trends in transport protocol development (e.g. QUIC, the RTCWEB data channel) that UDP encapsulation will prove a viable approach for deploying new

protocols in the Internet. This, however, leads us to a first problem that must be solved.

2.1. Problem Statement

UDP is often blocked by firewalls, or only enabled for a few well-known applications (e.g. DNS, NTP). Recent measurement work has shown that somewhere between 4% and 8% of Internet hosts may be affected by UDP impairment, depending on the population studied. Some networks (e.g. enterprise networks behind corporate firewalls) are far more likely to block UDP than others (e.g. residential wireline access networks).

In addition, some network operators assume that UDP is not often used for high-volume traffic, and is often a source of spoofing or reflected attack traffic, and is therefore safe to block or route-limit. This assumption is becoming less true than it once was: the volume of (good) UDP traffic is growing, mostly due to voice and video (real-time) services (e.g. RTCWEB) where TCP is not suitable.

Even if firewall vendors and administrators are willing to change firewall rules to allow more diverse UDP services, it is hard to track session state for UDP traffic. As UDP is unidirectional, it is unknown whether the receiver is willing to accept the connection. Further there is no way to figure how long state must be maintained once established. To efficiently establish state along the path we need an explicit contract, as is done implicitly with TCP today.

2.2. Information Exposed

To maintain state in the network, it must be possible to easily assign each packet to a session that is passing a certain network node. This state should be bound to something beyond the five-tuple to link packets together. In [I-D.trammell-spud-req], we propose the use of identifiers for groups of packets, called ("tubes"). This allows for differential treatment of different packets within one five-tuple flow, presuming the application has control over segmentation and can provide requirements on a per-tube basis. Tube IDs must be hard to guess: a tube ID in addition to a five-tuple as an identifier, given significant entropy in the tube ID, provides an additional assurance that only devices along the path or devices cooperating with devices along the path can send packets that will be recognized by middleboxes and endpoints as valid.

Further, to maintain state, the sender must explicitly indicate the start and end of a tube to the path, while the receiver must confirm connection establishment. This, together with the first packet following the confirmation, provides a guarantee of return

routability; i.e. that the sender is actually at the address it says it is. This implies all SPUD tubes must be bidirectional, or at least support a feedback channel for this confirmation. Even though UDP is not a bidirectional transport protocol, often services on top of UDP are bidirectional anyway. Even if not, we only require one packet to acknowledge a new connection. This is low overhead for this basic security feature. This connection set-up should not impose any additional start-up latency, so the sender must be also able to send payload data in the first packet.

If a firewall blocks a SPUD packet, it can be beneficial for the sender to know why the packet was blocked. Therefore a SPUD-aware middlebox should be able to send error messages. Such an error message can either be sent directly to the sender itself, or alternatively to the receiver that can decide to forward the error message to a sender or not.

2.3. Mechanism

A firewall or middlebox can use the tube ID as an identifier for its session state information. If the tube ID is large enough it will be hard for a non- eavesdropping attacker to guess the ID.

If a firewall receives a SPUD message that signals the start of a connection, it can decide to establish new state for this tube. Alternatively, it can also forward the packet to the receiver and wait if the connection is wanted before establishing state. To not require forwarding of unknown payload, a firewall might want to forward the initial SPUD packet without payload and only send the full packet if the connection has be accepted by the receiver.

The firewall must still maintain a timer to delete the state of a tube if no packets were received for a while. However, if a end signal is received the firewall can remove the state information faster.

If a firewall receives a SPUD message which does not indicate the start of a new tube and no state is available for this tube, it may decide to block the traffic. This can happen if the state has already timed out or if the traffic was rerouted. In addition a firewall may send an error message to the sender or the receiver indicating that no state information is available. If the sender receives such a message it can resend a start signal (potentially together with other tube state information) and continue its transmission.

2.4. Deployment Incentives

The ability to use existing firewall management best practices with new transport services over SPUD is necessary to ensure the deployability of SPUD. In today's Internet, application developers really only have two choices for transport protocols: TCP, or transports implemented at the application layer and encapsulated over UDP. SPUD provides a common shim layer for the second case, and the firewall traversal facility it provides makes these transports more likely to deploy.

It is not expected that the information provided by SPUD will enable all generic UDP-encapsulated transports to safely pass firewalls. However, it does make state handling easier for new services that a firewall administrator is willing to allow.

2.5. Security, Privacy, and Trust

The tube ID is scoped to the five-tuple. While this makes the tube ID useless for session mobility, it does mean that the valid ID space is sufficiently sparse to maintain the "hard to guess" property, and prevents tube IDs from being misused to track flows from the same endpoint across multiple addresses. This limitation may need further discussion.

By providing information about connection setup, SPUD exposes information equivalent to that available in the TCP header. It makes connection lifetime information explicit and accessible without specific higher-layer/application- level knowledge.

3. On-Path State Lifetime Discovery and Management

Once the problem of connection setup is solved, the problem arises of managing the lifetime of state associated with that connection at various devices along the path: NAT and stateful firewall state timeouts are a common cause of connectivity issues in the Internet.

3.1. Problem Statement

Devices along the path that must keep state in order to function cannot assume that signals tearing down a connection are provided reliably. This is also the case for current TCP traffic. Therefore, all stateful on-path devices must implement a mechanism to remove the state if no traffic is seen for a given flow or tube for a while. Usually this is implemented by maintaining a timeout since the last observed packet.

If the timeouts are set too low, on-path state might be discarded while the endpoint connection is still alive; in the case of firewalls and NATs, this can lead to unreliable connectivity. The common solution to this problem is for applications or transport protocols that do not have any productive traffic to send to send "heartbeat" or "keep-alive" packets to reset the state timeout along the path. However, since the minimum timeout along the path is unknown to the endpoint, implementers of transport and application . A default value of 150ms is commonly used today. This represents a fairly rapid generation of nonproductive traffic, and is especially onerous on battery- powered mobile devices, which must wake up radios and switch to a higher-power mode to transmit these nonproductive packets, leading to suboptimal power usage and shorter battery life.

3.2. Information Exposed

SPUD can be used to request that SPUD-aware middleboxes along the path expose their minimum state timeout value. Here, the sending endpoint sends a "accumulate minimum timeout" request along with some scratch space for middleboxes to place their timeout information in. Each middlebox inspects this value, and writes its own timeout only if lower than the present value.

Applications may also send a "timeout proposal" to devices along the path using a SPUD declaration that a given tube will send a packet at least once per interval, and if no packet is seen within that interval, it is safe to tear down state.

These two declarations may be used together, with middleboxes willing to use the application's value setting their timeouts on a per-tube basis, or exposing a lower timeout value to allow the application to adjust.

3.3. Mechanism

If a SPUD-aware middlebox that uses a timeout to clean up per-tube state receives a SPUD minimum timeout accumulation, it should expose its own timeout value if smaller than the one already given. Alternatively, if a value is already given, it might decide to use the given value as timeout for the state information of this tube. An endpoint receiving an accumulated minimum timeout should send it back to its remote peer via a feedback channel. Timeouts on each direction of a connection between two endpoints may, of course, be different, and are handled separately by this mechanism.

If a SPUD-aware middlebox that uses a timeout to clean up per-tube state receives a timeout proposal, it should set its timeout accordingly, subject to its own policy and configuration.

These mechanisms are of course completely advisory: there may be non-SPUD aware middleboxes on path which will ignore any proposed timeout and not expose their timeout information, and middleboxes must be configured with maximum timeout proposal they will accept in order to defend against state exhaustion attacks.

Endpoints must therefore be combine the use of these signals endpoint with a dynamic timeout discovery and adaptation mechanism, which uses the signals to set initial guesses as to the path timeout.

3.4. Deployment Incentives

Initially, if not widely deployed, there will be not much benefit to using this extension.

However, we can assume that there are usually only a small number of middleboxes on a given network path that hold per-tube state information. Endpoints have an incentive to request minimum timeout and to propose timeouts to improve convergence time for dynamic timeout adaptation mechanisms, and middleboxes have an incentive to cooperate to improve reliability of connections as well as state management. It is therefore likely that if information is exposed by a middlebox, this information is correct and can be used.

The more SPUD gets deployed, the more often endpoints will be able to set the heartbeat interval correctly. This will reduce the amount of unproductive traffic as well as the number of reconnections that cause additional latency.

Likewise, SPUD-aware middleboxes that expose timeout information are able to handle timeouts more flexibly, e.g. announcing lower timeout values when they have less space available for new state. Further if an endpoint announces a low pre-set value because the endpoint knows that it will only have short idle periods, the timeout interval could be reduced.

3.5. Security, Privacy, and Trust

Timeout proposals increase the risk of state exhaustion attacks for SPUD-aware middleboxes that naively follow them. Likewise, accumulated minimum timeouts could be used by malicious middleboxes to induce floods of useless heartbeat traffic along the path, and/or exhaust resources on endpoints that naively follow them. All timeout proposals and minimum timeouts must therefore be inputs to a dynamic timeout selection process, both at endpoints and on-path devices, which use these signals as hints but clamp their timeouts to sane values set by local policy.

While device timeout and heartbeat interval are generally not linked to privacy-sensitive information, a timeout proposal may add a number of bits of entropy to an endpoint's unique fingerprint. It is therefore advisable to suggest a small number of useful timeout proposals, in order to reduce this value's contribution to an endpoint fingerprint.

4. Path MTU Discovery

Similar to the state timeout problem is the Path MTU problem: differing MTUs on different devices along the path can lead to fragmentation or connectivity issues. This problem is made worse by the increasing proliferation of tunnels in the Internet, which reduce the MTU by the amount required for tunnel headers.

4.1. Problem Statement

In order to efficiently send packets along a path end to end, they must be sized to fit in the MTU of the "narrowest" link along the path. Algorithms for path MTU discovery have been defined and standardized for a quarter century, in [RFC1191] for IPv4 and [RFC1981] for IPv6, but they are not often implemented due in part to widespread impairment of ICMP. Packetization Layer Path MTU Discovery [RFC4821] (PLPMTUD) is a more recent attempt to solve the problem, which has the advantage of being transport-protocol independent and functional without ICMP feedback. SPUD, as a shim between UDP and superstrate transport protocols, is at the right place in the stack to implement PLPMTUD, and explicit cooperation can enhance its operation.

4.2. Information Exposed

SPUD can be used to request that SPUD-aware middleboxes along the path expose their next-hop path MTU value. Here, the sending endpoint sends a "accumulate minimum MTU" request along with some scratch space for middleboxes to place the next-hop MTU for the given tube. Each middlebox inspects this value, and writes its own next-hop MTU only if lower than the present value.

A SPUD-aware middlebox that receives a packet that is too big for the next-hop MTU can send back a signal associated with the tube directly to the sender, including the next-hop MTU.

4.3. Mechanism

PLPMTUD functions by dynamically increasing the size of packets sent, and reacting to the loss of the first "too large" packet as an MTU reduction signal, instead of a congestion signal. This must be

implemented in cooperation with the superstrate transport protocol, as it is responsible for how non-MTU-related loss is treated.

When an endpoint receives an accumulated minimum MTU, it should should send it back to its remote peer via a feedback channel. The minimum of this value and any direct next-hop MTU signals received from SPUD-aware middleboxes can be used as a hint to the sender's PLPMTUD process, as a likely upper bound for path MTU associated with a tube.

4.4. Deployment Incentives

As with state lifetime discovery, these signals are of little initial utility to endpoints before SPUD-aware middleboxes are deployed. However, SPUD-aware middleboxes that sit at potential MTU breakpoints along a path, either those which terminate tunnels or bridge networks with two different link types, have an incentive to improve reliability by responding to accumulation requests and sending next-hop MTU messages to SPUD-aware endpoints.

4.5. Security, Privacy, and Trust

As with state lifetime discovery, Minimum MTU and next-hop MTU signals could be used by malicious middleboxes to set the endpoint's maximum packet size to inefficiently small sizes, if the endpoint follows them naively. For that reason, endpoints should use this information only as hints to improve the operation of PLPMTUD, and may probe above the value derived from the SPUD- supplied information when deemed appropriate by endpoint policy or transport protocol requirements.

5. Low-Latency Service

5.1. Problem Statement

Networks are often optimized for low loss rates and high throughput by providing large buffers that can absorb traffic spikes and rate variations while holding enough data to keep the link full. This is beneficial for applications like high-priority bulk transfer, where only the total transfer time is of interest. High-volume interactive applications, such as videoconferencing, however, have very different requirements. Usually these applications can tolerate higher loss rates, while having hard latency requirements.

Large network buffers may induce high queuing delays due to cross traffic using loss-based congestion control, which must periodically fill the buffer to induce loss during probing for additional bandwidth. This queueing delay can negatively impact the quality of

experience for competing interactive applications, even making them unusable.

5.2. Information Exposed

The simplest mechanism for solving this problem is to separate loss-sensitive from latency-sensitive traffic, as proposed using DSCP codepoints in [I-D.you-tsvwg-latency-loss-tradeoff]. This signal could also be emitted as a per-packet signal within SPUD, since DSCP codepoints are often used for internal traffic engineering and therefore cleared at network borders. This indication does not prioritize one kind of traffic over the other: while loss- sensitive traffic might face larger buffer delay but lower loss rate, latency-sensitive traffic has to make exactly the opposite tradeoff.

An endpoint can also indicate a maximum acceptable single-hop queueing delay per tube, expressed in milliseconds. While this mechanism does not guarantee that sent packets will experience less than the requested delay due to queueing delay, it can significantly reduce the amount of traffic uselessly sitting in queues, since at any given instance only a small number of queues along a path (usually only zero or one) will be full.

5.3. Mechanism

A middlebox may use the loss-/latency tradeoff signal to assign packet to the appropriate type of service, if different services are implemented at this middlebox. Traffic not indicating a low loss or low latency preference would still be assigned to today's best-effort service, while a new low latency service would be introduced in addition.

The simplest implementation of such a low latency service (without disturbing existing traffic) is to manage traffic with the latency-sensitive flag set in a separate queue. This queue either, in itself, provides only a short buffer which induces a hard limit for the maximum (per-queue) delay or uses an AQM (such as PIE/CoDel) that is configured to keep the queuing delay low.

In such a two-queue system the network provider must decide about bandwidth sharing between both services, and might or might not expose this information. Initially there will only be a few flows that indicate low latency preference. Therefore at the beginning this service might have a low maximum bandwidth share assigned in the scheduler. However, the sharing ratio should be adapted to the traffic load/number of flows in each service class over long timescales.

Applications and endpoints setting the latency sensitivity flag on a tube must be prepared to experience relatively higher loss rates on that tube, and should use techniques such as Forward Error Correction (FEC) to cope with these losses.

If a maximum per-hop delay is indicated by the sender, a SPUD- aware router might drop any packet which would be placed in a queue that has more than the maximum single-hop delay at that point in time before queue admission. Thereby the overall congestion can be reduced early instead of withdrawing the packet at the receiver after it has blocked network resources for other traffic.

A transport protocol at an endpoint indicating the maximum per-hop delay must be aware that is might face higher loss rates under congestion than competing traffic on the same bottleneck.

5.4. Deployment Incentives

Application developers go to a great deal of effort to make latency-sensitive traffic work over today's Internet. However, if large delays are induced by the network, an application at the endpoint cannot do much. Therefore applications can benefit from further support by the network.

Network operators have already realized a need to better support low latency services. However, they want to avoid any service degradation for existing traffic as well as risking stability due to large configuration changes. Introducing an additional service for latency-sensitive traffic that can exist in parallel to today's network service helps this problem.

5.5. Security, Privacy, and Trust

An application cannot benefit from wrongly indicating loss- or latency- sensitivity, as it has to make a tradeoff between low loss and potential high delay or low delay and potential high loss.

A simple classification of traffic as loss- or latency-sensitive does not expose privacy-critical information about the user's behavior; indeed, it exposes far less than presently used by DPI-based traffic classifiers that would be used to determine the latency sensitivity of traffic passing a middlebox.

6. Reordering Sensitivity

6.1. Problem Statement

TCP's fast retransmit mechanism interprets the reception of three duplicated acknowledgement (where the acknowledgement number is the same than in the previous acknowledgement) as a signal for loss detection. However, a missing packet in the sequence number space must not always be lost. Simple reordering where one packet takes a longer path than (at least three) subsequent packets can have the same effect.

In addition in TCP, loss is an implicit signal for network congestion. Therefore the reception of three duplicated acknowledgement will cause a TCP sender to reduce its sending rate. To avoid unnecessary performance decreases, today's in-network mechanisms usually aim to avoid reordering. However, this complicates these mechanism significantly and usually requires per-flow state, e.g. in case of Equal Cost Multipath (ECMP) routing where a hash of the 5 tuple would need to be mapped to the right path.

Even though the majority of traffic in the Internet is still TCP, it is likely that new protocols will be design such that they are (more) robust to reordering. Further with an increasing deployment of ECN, even TCP's congestion control reaction based on duplicated acknowledgements could be relaxed (e.g. by reducing the sending rate gradually depending on the number of lost packets).

However, as middlebox can not know if a certain traffic flow is sensitive to reordering or not, they have to treat all traffic as equally and try to always avoid reordering. (This does not only complicate these mechanism but might also block the deployment of new services.)

6.2. Information Exposed

Reordering-sensitivity is a per tube signal (as reordering can only happen with a flow multiple packets). However, to avoid state in middlebox, it would be beneficial to have a reordering-sensitive flag in each packet.

A transport should set the bit if it is not sensitive to reordering, e.g. if it uses a more advance mechanism (than duplicated acknowledgement) for loss detection, or if the congestion control reaction to this signal imposes only a small performances penalty, or if the flow is short enough that it will not impact its performance.

6.3. Mechanism

A middlebox that implement an in-network function that could lead to varying end-to-end delay and reordering (as packets might overtake each other on different paths or within the network device), do not need to perform any additional action if the reordering-sensitivity flag is not set. However, if the flag is set, the middlebox should avoid reordering by e.g. holding per- tube state and make sure that all packets belonging to the same tube will not be re-ordered.

6.4. Deployment Incentives

Today by default middlebox assume that all traffic is reordering-sensitive which complicates certain in-network mechanism or might also block the deployment of new services. If a middlebox would know that certain traffic is not reordering-sensitive, it could reduce state, speed-up processing, or even implement new services.

Applications that are not loss-sensitive (because they e.g. uses FEC) usually are also not reordering-sensitive. At the same time these application are often sensitive to latency. If the transport handles reordering appropriately and signal this semantic information to the network, the appropriate network treatment can likely also result in lower end-to-end or at least enables the network device to impose any additional delay (e.g. to set up state) on these packets.

6.5. Security, Privacy, and Trust

No trust relationship is needed as the provided information do not results in a preferential treatment. Only transport semantics are exposed that to not contain any private information.

7. Application-Limited Flows

7.1. Problem Statement

Many flows are application-limited, where the application itself adapts the limit to changing traffic conditions or link characteristics, such as with unicast adaptive bitrate streaming video. This adaptation is difficult, since TCP cross-traffic will often probe for available bandwidth more aggressively than the application's control loop. Further complicating the situation is the fact that rate adaptation may have negative effects on the user's quality of experience, and should therefore be done infrequently.

7.2. Information Exposed

A SPUD endpoint sending application-limited traffic can provide an explicit per-tube indication of the maximum intended data rate needed by the current encoding or data source. If the bottleneck device is SPUD-aware, it can use this information to decide how to correctly treat the tube, e.g. setting a rate limit or scheduling weight if served from its own queue.

A SPUD endpoint could also send a "minimum rate limit accumulation" request, similar to the other accumulation requests outlined above, where SPUD-aware routers and middleboxes could note the maximum bandwidth available to a tube. Receiving this signal on a feedback channel could allow a sender to more quickly adapt its sending rate. This rate limit information might be derived from local per-flow or per-tube rate limit policy, as well as from current information about load at the router.

These signals can be sent throughout the lifetime of the flow, to help adapt to changing application demands and/or network conditions.

7.3. Mechanism

Maximum expected data rate exposed by the endpoints could be used to make routing decisions and queue selection decisions at SPUD-aware routers, if different paths or queues with different capacity, delay, and load characteristics are available.

A SPUD-aware router that indicates a rate limit can be used by the sender to choose an encoding. However, the sender should still implement a mechanism to probe for available bandwidth to verify the provided information. As a certain rate limit is expected, the sender should probe carefully around this rate.

These mechanisms can also be used for rate increases. If a sender receives an indication that more bandwidth is available it should probe carefully, instead of switching to the higher rate immediately, and decrease its sensitivity to loss (e.g. through the use of additional FEC) which will provide additional protection as soon as the new capacity limit is reached. Likewise, a SPUD- aware router that receives an indication that a flow intends to increase its might prioritize this flow for a certain (short) time to enable a smoother transition.

7.4. Deployment Incentives

Endpoints that indicate maximum sending rate for application-limited traffic on SPUD-aware networks allow the operators of those networks to better handle traffic. This can benefit the service quality and increase the user's satisfaction with the provided network service.

Currently applications have no good indication when to change their coding rate. Rate increases are especially hard. Further, frequent rate changes should be avoided for quality of experience. Cooperative indication of intended and available sending rate for application-limited flows can simplify probing, and provide signals beyond loss to react effectively to congestion.

7.5. Security, Privacy, and Trust

Both endpoints and SPUD-aware middleboxes should react defensively to rate limit and rate intention information. Endpoints and middleboxes should use measurement and probing to verify that rate information is accurate, but the exposed rate information can be used as hints to routing, scheduling, and rate determination processes.

8. Priority Multiplexing

8.1. Problem Statement

Many services require multiple parallel transmissions to transfer different kinds of data which have clear priority relationships among them. For example, in WebRTC, audio frames should be prioritized over video frames. Sometimes these transmissions happen in different flows, and sometimes some packets within a flow have higher priority than others, for example I-frames in video transmissions. However, current networks will treat all packets the same in case of congestion and might e.g. drop audio packets while video and control traffic are still transmitted.

8.2. Information Exposed

A SPUD sender may indicate a that one tube should "yield" to another, i.e. that it should have lower relative priority than another tube in the same flow. Similarly, individual packets within a tube could be marked as having lower priority. This information can be used to preferentially drop less important packets e.g. carrying information that could be recovered by FEC.

With a stronger integration of codec and transport protocols, SPUD could even indicate more fine-grained priority levels to provide automatic graceful degradation of service within the network itself.

8.3. Mechanism

Designing a general-purpose mechanism that maps relative priorities from the yield information exposed via SPUD to correct per-tube and per-packet treatment at any point in the Internet, is an extremely hard problem and a possible subject for future research. It appears impossible at this writing to design a straightforward mapping function from these relative priorities per- flow to absolute priorities across flows in a fair way.

However, in the not-uncommon case that exists in many access networks, where the bottleneck link has per-user queues and can enforce per-user fairness, the relative priorities can be mapped to absolute priorities, and simple priority queueing at the bottleneck can be used. Lower priority packets within a tube, however, should be assigned to the tube's priority class, and preferentially dropped instead, e.g. using a different drop threshold at the queue.

8.4. Deployment Incentives

Deployment incentives for priority multiplexing are similar to those for bandwidth declaration for app-limited flows as in Section 7.4: endpoints that correctly declare priority information will experience better quality of service on SPUD-enabled networks, and SPUD-enabled networks get information that allows them to better manage traffic.

8.5. Security, Privacy, and Trust

Since yield information can only be used to disadvantage an application's traffic relative to its own traffic, there is no incentive for applications to declare incorrect yielding.

The pattern and relative volume of traffic in different yield classes may be used to "fingerprint" certain applications, though it is not clear whether this provides additional information beyond that contained inter-packet delay and volume patterns.

9. In-Band Measurement

9.1. Problem Statement

The current Internet protocol stack has very limited facilities for network measurement and diagnostics. The only explicit measurement feature built into the stack is ICMP Echo ("ping"). In the meantime, the Internet measurement community has defined many inference- and assumption-based approaches for getting better information out of the network: traceroute and BGP looking glasses for topology information, TCP sequence number and TCP timestamp based approaches for latency

and loss estimation, and so on. Each of these uses values placed on the wire for the internal use of the protocol, not for measurement purposes, and do not necessarily apply to the deployment of new protocols or changes to the use of those values by protocol implementations. Approaches involving the encryption of transport protocol and application headers (indeed, including that the authors advance in [I-D.trammell-spud-req]) will break most of these, as well.

Replacing the information used for measurement with values defined explicitly to be used for measurement in a transport protocol independent way allows explicit endpoint control of measurability and measurement overhead.

We note that current work in IPPM [I-D.ietf-ippm-6man-pdm-option] proposes a roughly equivalent, IPv6-only, kernel-implementation-only facility.

9.2. Information Exposed

The "big five" metrics - latency, loss, jitter, data rate / goodput, and reordering - can be measured using a relatively simple set of primitives. Packet receipt acknowledgment using a cumulative nonce echo allows both endpoint and on-path measurement of loss and reordering as well as goodput (when combined with layer 3 packet length headers). A timestamp echo facility, analogous to TCP's timestamp option but using an explicitly defined, constant-rate clock and exposure of local delta (time between receipt and subsequent transmission).

The cumulative nonce echo consists of two values: a number identifying a given packet (nonce), which also identifies all retransmissions of the packet, and a number which is the sum of all packet identifiers received from the remote endpoint (echo), modulo the maximum value of the echo field. Nonces need not be sequential, or even monotonic, but two packets with the same nonce should not be simultaneously in flight. These are exposed on a per-packet basis, but need not appear on every packet in the tube or flow, with the caveat that lower sampling rates lead to lower sensitivity.

The timestamp echo consists of three values: The time in terms of ticks of a constant rate clock that a packet is sent, the echo of the last such timestamp received from the remote endpoint, and the number of ticks of the sender's clock between the receipt of the last timestamp from the remote endpoint and the transmission of the packet containing the echo. This last delta value is the missing link in TCP sequence number based and timestamp option based latency estimation.

The information exposed is roughly equivalent than that currently exposed by TCP as a side effect of its operation, but defined such that they are explicitly useful for measurement, useful regardless of transport protocol, and such that information exposure is in the explicit control of the endpoint (when the superstrate transport protocol's headers are encrypted).

9.3. Mechanism

The nonce and timestamp echo information, emitted as per-packet signals in the SPUD header, can be used by any device which can see it to estimate performance metrics on a per-tube basis. This includes both remote endpoints, as well as passive performance measurement devices colocated with network gateways.

9.4. Deployment Incentives

Initial deployment of this facility is most likely in closed networks such as enterprise data centers, where a single administrative entity owns the network and the endpoints, can control which flows and tubes are annotated with measurement information, and can benefit from the additional insight given during network troubleshooting by explicit measurement headers.

Further, since the provided measurement information is exposed by SPUD to the far-endpoint, it can be used for performance enhancement on these layers. Once the facility is deployed in SPUD-aware endpoints, it can also be used for inter-network and cross-Internet performance measurement and debugging (replacing today's processing-intensive DPI mechanisms).

9.5. Security, Privacy, and Trust

The cumulative nonce and timestamp echo leaks no more information about the traffic than the TCP header does. Indeed, since the cumulative nonce does not include sequence number information or other protocol-internal information, it allows passive measurement of loss and latency without giving measurement devices access to information they could use to spoof valid packets within a transport layer connection.

In order to prevent middleboxes from modifying measurement-relevant information, these per-packet signals will need to be integrity protected by SPUD.

Performance measurement boxes at gateways which observe and aggregate these signals will necessarily need to trust their accuracy, but can

verify their plausibility by calculating nonce sums and synchronizing timing clocks.

10. IANA Considerations

This document has no actions for IANA.

11. Security Considerations

Security and privacy considerations for each use case are given in the corresponding Security, Privacy, and Trust subsection.

12. Acknowledgments

This document grew in part out of discussions of initial use cases for middlebox cooperation at the IAB SEMI Workshop and the IETF 92 SPUD BoF; thanks to the participants. Some use case details came out of discussions with the authors of the [I-D.trammell-spud-req]: in addition to the editors of this document, David Black, Ken Calvert, Ted Hardie, Joe Hildebrand, Jana Iyengar, and Eric Rescorla. Section 9 is based in part on discussions and ongoing work with Mark Allman and Rob Beverly.

This work is supported by the European Commission under Horizon 2020 grant agreement no. 688421 Measurement and Architecture for a Middleboxed Internet (MAMI), and by the Swiss State Secretariat for Education, Research, and Innovation under contract no. 15.0268. This support does not imply endorsement.

13. Informative References

[I-D.hildebrand-spud-prototype]
Hildebrand, J. and B. Trammell, "Substrate Protocol for User Datagrams (SPUD) Prototype", draft-hildebrand-spud-prototype-03 (work in progress), March 2015.

[I-D.ietf-ippm-6man-pdm-option]
Elkins, N. and M. Ackermann, "IPv6 Performance and Diagnostic Metrics (PDM) Destination Option", draft-ietf-ippm-6man-pdm-option-01 (work in progress), October 2015.

[I-D.trammell-spud-req]
Trammell, B. and M. Kuehlewind, "Requirements for the design of a Substrate Protocol for User Datagrams (SPUD)", draft-trammell-spud-req-02 (work in progress), March 2016.

[I-D.you-tsvwg-latency-loss-tradeoff]
You, J., Welzl, M., Trammell, B., Kuehlewind, M., and K. Smith, "Latency Loss Tradeoff PHB Group", draft-you-tsvwg-latency-loss-tradeoff-00 (work in progress), March 2016.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <http://www.rfc-editor.org/info/rfc1191>.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <http://www.rfc-editor.org/info/rfc1981>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <http://www.rfc-editor.org/info/rfc4821>.

Authors' Addresses

Mirja Kuehlewind (editor)
ETH Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Email: mirja.kuehlewind@tik.ee.ethz.ch

Brian Trammell (editor)
ETH Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Email: ietf@trammell.ch