Requirements for the design of a Substrate Protocol for User Datagrams (SPUD)
draft-trammell-spud-req-02

Abstract

   The Substrate Protocol for User Datagrams (SPUD) BoF session at the
   IETF 92 meeting in Dallas in March 2015 identified the potential need
   for a UDP-based encapsulation protocol to allow explicit cooperation
   with middleboxes while using new, encrypted transport protocols.
   This document proposes an initial set of requirements for such a
   protocol, and discusses tradeoffs to be made in further refining
   these requirements.

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at http://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on September 12, 2016.

Table of Contents

1. Motivation

   A number of efforts to create new transport protocols or experiment
   with new network behaviors in the Internet have been built on top of
   UDP, as it traverses firewalls and other middleboxes more readily
   than new protocols do. Each such effort must, however, either manage
   its flows within common middlebox assumptions for UDP or train the
   middleboxes on the new protocol (thus losing the benefit of using
   UDP). A common Substrate Protocol for User Datagrams (SPUD) would
   allow each effort to re-use a set of shared methods for notifying
   middleboxes of the flows' semantics, thus avoiding both the
   limitations of current flow semantics and the need to re-invent the
   mechanism for notifying the middlebox of the new semantics.

   As a concrete example, it is common for some middleboxes to tear down
   required state (such as NAT bindings) very rapidly for UDP flows. By
   notifying the path that a particular transport using UDP maintains
   session state and explicitly signals session start and stop using the
   substrate, the using protocol may reduce or avoid the need for
   heartbeat traffic.

   This document defines a specific set of requirements for a SPUD
   facility, based on analysis on a target set of applications to be
   developed on SPUD developing experience with a prototype described in
   [I-D.hildebrand-spud-prototype]. It is intended as the basis for
   determining the next steps to make progress in this space, including
   possibly chartering a working group for specific protocol engineering
   work.

   Within this document, requirements are presented as for a facility
   implementable as an encapsulation protocol, atop which new transports
   ("superstrates") can be built. Alternately, these could be viewed as
   a set of requirements for future transport protocol development
   without a layer separation between the transport and the superstrate.

   The final intention of this work is to make it possible to define and
   deploy new transport protocols that use encryption to protect their
   own operation as well as the confidentiality, authenticity,
   integrity, and linkability resistance of their payloads. The
   accelerating deployment of encryption will render obsolete network
   operations techniques that rely on packet inspection and modification
   based upon assumptions about the protocols in use. This work will

allow the replacement the current regime of middlebox inspection and
modification of transport and application- layer headers and payload
with one that allows inspection only of information explicitly
exposed by the endpoints, and modification of such information only
under endpoint control.

2. History

An outcome of the IAB workshop on Stack Evolution in a Middlebox
Internet (SEMI) [RFC7663], held in Zurich in January 2015, was a
discussion on the creation of a substrate protocol to support the
deployment of new transport protocols in the Internet. Assuming that
a way forward for transport evolution in user space would involve
encapsulation in UDP datagrams, the workshop noted that it may be
useful to have a facility built atop UDP to provide minimal signaling
of the semantics of a flow that would otherwise be available in TCP.
At the very least, indications of first and last packets in a flow
may assist firewalls and NATs in policy decision and state
maintenance. This facility could also provide minimal application-
to- path and path-to-application signaling, though there was less
agreement about what should or could be signaled here. Further
transport semantics would be used by the protocol running atop this
facility, but would only be visible to the endpoints, as the
transport protocol headers themselves would be encrypted, along with
the payload, to prevent inspection or modification. This encryption
might be accomplished by using DTLS [RFC6347] as a subtransport
[I-D.huitema-tls-dtls-as-subtransport] or by other suitable methods.

The Substrate Protocol for User Datagrams (SPUD) BoF was held at IETF
92 in Dallas in March 2015 to develop this concept further. It is
clear from discussion before and during the SPUD BoF that any
selective exposure of traffic metadata outside a relatively
restricted trust domain must be advisory, non-negotiated, and
declarative rather than imperative. This conclusion matches
experience with previous endpoint-to-middle and middle-to-endpoint
signaling approaches. As with other metadata systems, exposure of
specific elements must be carefully assessed for privacy risks and
the total of exposed elements must be so assessed. Each exposed
parameter should also be independently verifiable, so that each
entity can assign its own trust to other entities. Basic transport
over the substrate must continue working even if signaling is ignored
or stripped, to support incremental deployment. These restrictions
on vocabulary are discussed further in
[I-D.trammell-stackevo-explicit-coop]. This discussion includes
privacy and trust concerns as well as the need for strong incentives
for middlebox cooperation based on the information that are exposed.

3. Terminology

   This document uses the following terms:

   o Superstrate: : The transport protocol or protocol stack "above"
      SPUD, that uses SPUD for explicit path cooperation and path
      traversal. The superstrate usually consists of a security layer
      (e.g. TLS, DTLS) and a transport protocol, or a transport
      protocol with integrated security features, to protect headers and
      payload above SPUD.

   o Endpoint: : One end of a communication session, located on a
      single node that is a source or destination of packets in that
      session. In this document, this term may refer to either the SPUD
      implementation at the endpoint, the superstrate implementation
      running over SPUD, or the applications running over that
      superstrate.

   o Path: : The sequence of Internet Protocol nodes and links that a
      given packet traverses from endpoint to endpoint.

   o Middlebox: : As defined in [RFC3234], a middlebox is any
      intermediary device performing functions other than the normal,
      standard functions of an IP router on the datagram path between a
      source host and destination host; e.g. making decisions about
      forwarding behavior based on other than addressing information,
      and/or modifying a packet before forwarding.

4. Use Cases

   The primary use case for endpoint to path signaling in the Internet,
   making use of packet grouping, is the binding of limited related
   semantics (start, ack, and stop) to a flow or a group of packets
   within a flow which are semantically related in terms of the
   application or superstrate. By explicitly signaling start and stop
   semantics, a flow allows middleboxes to use those signals for setting
   up and tearing down their relevant state (NAT bindings, firewall
   pinholes), rather than requiring the middlebox to infer this state
   from continued traffic. At best, this would allow the application to
   refrain from sending heartbeat traffic, which might result in reduced
   radio utilization and thus greater battery life on mobile platforms.

   SPUD could also be used to provide information relevant for network
   treatment for middleboxes as a replacement for deep packet inspection
   for traffic classification purposes, rendered ineffective by
   superstrate encryption. In this application, properties would be
   expressed in terms of network-relevant parameters (intended
   bandwidth, latency and loss sensitivity, etc.) as opposed to

application-relevant semantics. See
[I-D.trammell-stackevo-explicit-coop] for discussion on limitations
in signaling in untrusted environments.

SPUD may also provide some facility for SPUD-aware nodes on the path
to signal some property of the path relative to a tube to the
endpoints and other SPUD- aware nodes on the path. The primary use
case for path to application signaling is parallel to the use of ICMP
[RFC0792], in that it describes a set of conditions (including
errors) that applies to the datagrams as they traverse the path.
This usage is, however, not a pure replacement for ICMP but a
"5-tuple ICMP" for error messages which should be application-
visible; these would traverse NATs in the same way as the traffic
related to it, and be deliverable to the application with appropriate
tube information.

Link-layer characteristics of use to the transport layer (e.g.,
whether a high-transient-delay, highly-buffered link such as LTE is
present on the path) could also be signaled using this path-to-
endpoint facility.

Further use cases are outlined in more detail in
[I-D.kuehlewind-spud-use-cases].

5. Functional Requirements

The following requirements detail the services that SPUD must provide
to superstrates, endpoints, and middleboxes using SPUD.

5.1. Grouping of Packets (into "tubes")

Transport semantics and many properties of communication that
endpoints may want to expose to middleboxes are bound to flows or
groups of flows (five- tuples). SPUD must therefore provide a basic
facility for associating packets together (into what we call a
"tube", for lack of a better term) and associate information to these
groups of packets. Each packet in a SPUD "flow" (determined by
5-tuple) belongs to exactly one tube. Notionally, a tube consists of
a set of packets with a set of common properties, that should
therefore receive equivalent treatment from the network; these tubes
may or may not be related to separate semantic entities in the
superstrate (e.g. SCTP streams).

The simplest mechanisms for association involve the addition of an
identifier to each packet in a tube. Other mechanisms that don't
directly encode the identifier in a packet header, but instead
provide it in a way that it is simple to derive from other
information available in the packet at the endpoints and along the

path, are also possible. In any cases, for the purposes of this
requirement we treat this identifier as a simple vector of N bits.
The properties of the tube identifier are subject to tradeoffs on the
requirements for privacy, security, ease of implementation, and
header overhead efficiency.

In determining the optimal size and scope for this tube identifier,
we first assume that the 5-tuple of source and destination IP
address, UDP port, and IP protocol identifier (17 for UDP) is used in
the Internet as an existing flow identifier, due to the widespread
deployment of network address and port translation. We conclude that
SPUD tube IDs should be scoped to this 5-tuple.

While a globally-unique identifier would allow easier state
comparison and migration for mobility use cases, it would have two
serious disadvantages. First, N would need to be sufficiently large
to minimize the probability of collision among multiple tubes having
the same identifier along the same path during some period of time.
A 128-bit UUID [RFC4122] or an identifier of equivalent size
generated using an equivalent algorithm would probably be sufficient,
at the cost of 128 bits of header space in every packet. Second,
globally unique tube identifiers would also introduce new
possibilities for user and node tracking, with a serious negative
impact on privacy. We note that global identifiers for mobility,
when necessary to expose to the path, can be supported separately
from the tube identification mechanism, by using a generic tube-
grouping application-to-path signaling bound to the tube.

Even when tube IDs are scoped to 5-tuples, N must still be
sufficiently large, and the bits in the identifier sufficiently
random, that possession of a valid tube ID implies that a node can
observe packets belonging to the tube (see Section 6.6). This
reduces the chances of success of blind packet injection attacks of
packets with guessed valid tube IDs.

When scoped to 5-tuples, the forward and backward directions of a
bidirectional connection will have different tube IDs, since these
will necessarily take different paths and may interact with a
different set of middleboxes due to asymmetric routing. SPUD will
therefore require some facility to note that one tube is the
"reverse" direction of another, a general case of the tube grouping
signal above.

5.2. Endpoint to Path Signaling

SPUD must be able to provide information scoped to a tube from the
end- point(s) to all SPUD-aware nodes on the path about the packets

in that tube. We note that in-band signaling would meet this
requirement.

5.3. Path to Endpoint Signaling

SPUD must be able to provide information from a SPUD-aware middlebox
to the endpoint. This information is associated with a tube, in
terms of "the properties of the path(s) the packets in this tube will
traverse". Path-to- endpoint signaling must be made available to the
superstrate and/or the application at the endpoint. We note that in-
band signaling would meet this requirement.

5.4. Tube Start and End Signaling

SPUD must provide a facility for endpoints to signal that a tube has
started, that the start of the tube has been acknowledged and
accepted by the remote endpoint(s), and that a tube has ended and its
state can be forgotten by the path. Given unreliable signaling (see
Section 7.10) both endpoints and devices on the path must be
resilient to the loss of any of these signals. Specifically,
timeouts are still necessary to clean up stale state. See
Section 8.6 and Section 8.7 for more discussion on tube start and end
signaling.

5.5. Declarative signaling

All information signaled via SPUD is defined to be declarative (as
opposed to imperative). A SPUD endpoint must function correctly if
no middlebox along the path understands the signals it sends, or if
sent signals from middleboxes it does not understand. Likewise, a
SPUD-aware middlebox must function correctly if sent signals from
endpoints it does not understand, or in the absence of expected
signals from endpoints.

5.6. Extensibility

SPUD must enable multiple new transport semantics and application/
path declarations without requiring updates to SPUD implementations
in middleboxes.

The use of SPUD for experimental signaling must be possible either
without the registration of codepoints or namespaces with IANA, or
with trivially easy (First Come, First Served [RFC5226] registration
of such codepoints.

6. Security Requirements

6.1. Privacy

   SPUD must allow endpoints to control the amount of information
   exposed to middleboxes, with the default being the minimum necessary
   for correct functioning. This includes the cryptographic protection
   of transport layer headers from inspection by devices on path, in
   order to prevent ossification of these headers.

6.2. Authentication

   The basic SPUD protocol must not require any authentication or a
   priori trust relationship between endpoints and middleboxes to
   function. However, SPUD should interoperate with the presentation/
   exchange of authentication information in environments where a trust
   relationship already exists, or can be easily established, either in-
   band or out-of-band, and use this information where possible and
   appropriate.

   Given the advisory nature of the signaling it supports, SPUD may also
   support eventual authentication: authentication of a signal after the
   reception of a packet after that containing the signal.

6.3. Integrity

   SPUD must provide integrity protection of exposed information in
   SPUD- encapsulated packets, though the details of this integrity
   protection are still open; see Section 8.2.

   Endpoints should be able to detect changes to headers SPUD uses for
   its own signaling (whether due to error, accidental modification, or
   malicious modification), as well as the injection of packets into a
   SPUD flow (defined by 5-tuple) or tube by nodes other than the remote
   endpoints. Errors and accidental modifications can be detected using
   a simple checksum over the SPUD header, while detecting malicious
   modifications requires cryptographic integrity protection. Similar
   to Section 6.2, cryptographic integrity protection may also be
   eventual.

   Integrity protection of the superstrate is left up to the
   superstrate.

6.4. Encrypted Feedback

   Some use cases involve collecting information along a forward path
   from a sending endpoint to a receiving endpoint. In cases where this
   information is also useful to the sending endpoint, SPUD must provide

a feedback channel to communicate this information back to the
sender. As this information does not need to be exposed to the path,
this feedback channel should be encrypted for confidentiality and
authenticity, when available (see Section 6.2).

6.5. Preservation of Security Properties

The use of SPUD must not weaken the essential security properties of
the superstrate: confidentiality, integrity, authenticity, and
defense against linkability. If the superstrate includes payload
encryption for confidentiality, for example, the use of SPUD must not
allow deep packet inspection systems to have access to the plaintext.
Likewise, the use of SPUD must not create additional opportunities
for linkability not already existing in the superstrate.

With respect to access control, SPUD itself must not be used to
negotiate the means to lift administrative prohibition of certain
traffic, although it could be used to provide more useful information
why it is prohibited.

6.6. Proof a device is on-path

Devices may make assertions of network characteristics relevant to a
flow. One way these assertions can be assessed is by a demonstration
that the device making it is on-path to the flow and so could adjust
the characteristics to match the assertion. SPUD must therefore
allow endpoints to distinguish on- path devices from devices not on
the path. Network elements may also need to confirm that
application-to-path assertions are made by the source indicated in
the flow. In both cases, return routability (as in Section 6.7) may
offer one incrementally deployable method of testing the topology to
make this confirmation.

6.7. Protection against trivial abuse

Malicious background traffic is a serious problem for UDP-based
protocols due to the ease of forging source addresses in UDP together
with only limited deployment of network egress filtering [RFC2827].
Trivial abuse includes flooding and state exhaustion attacks, as well
as reflection and amplification attacks. SPUD must provide minimal
protection against this trivial abuse. This probably implies that
SPUD should provide:

o a proof of return routability, that the endpoint identified by a
   packet's source address receives packets sent to that address;

o a feedback channel between endpoints;

o a method to probabilistically discriminiate legitimate SPUD
   traffic from reflected malicious traffic; and

o mechanisms to protect against state exhaustion and other denial-
   of-service attacks.

We note that using a "magic number" or other pattern of bits in an
encapsulation-layer header not used in any widely deployed protocol
has the nice property that no existing node in the Internet can be
induced to reflect traffic containing it. This allows the magic
number to provide probabilistic assurance that a given packet is not
reflected, assisting in meeting this requirement.

7. Technical Requirements

The following requirements detail the constraints on how the SPUD
facility must meet its functional requirements.

7.1. Middlebox Traversal

SPUD, including all path-to-endpoint and endpoint-to-path signaling
as well as superstrate and superstrate payload, must be able to
traverse middleboxes and firewalls, including those that are not
SPUD-aware. Therefore SPUD must be encapsulated in a transport
protocol that is known to be accepted on a large fraction of paths in
the Internet, or implement some form of probing to determine in
advance which transport protocols will be accepted on a certain path.
This encapsulation will require port numbers to support endpoints
connected via network address and port translation (NAPT). We note
that UDP encapsulation would meet these requirements.

7.2. Low Overhead in Network Processing

SPUD must be low-overhead, specifically requiring very little effort
to recognize that a packet is a SPUD packet and to determine the tube
it is associated with. We note that a magic number as in Section 6.7
would also have a low probability of colliding with any non- SPUD
traffic, therefore meeting the recognition requirement. Tube
identifiers appearing directly in the encapsulation-layer header
would meet the tube association requirement.

7.3. Implementability in User-Space

To enable fast deployment SPUD and superstrates must be implementable
without requiring kernel replacements or modules on the endpoints,
and without having special privilege (such as is required for raw
packet transmission, i.e. root or "jailbreak") on the endpoints. We
note here that UDP would meet this requirement, as nearly all

operating systems and application development platforms allow a
userspace application to open UDP sockets.

7.4. Incremental Deployability in an Untrusted, Unreliable Environment

SPUD must operate in the present Internet. In order to maximize
deployment, it should also be useful between endpoints even before
the deployment of middleboxes that understand it. The information
exposed over SPUD must provide incentives for adoption by both
endpoints and middleboxes, and must maximize privacy (by minimizing
information exposed). Further, SPUD must be robust to packet loss,
duplication and reordering by the underlying network service. SPUD
must work in multipath, multicast, and endpoint multi- homing
environments.

Incremental deployability likely requires limitations of the
vocabulary used in signaling, to ensure that each actor in a non-
trusted environment has incentives to participate in the signaling
protocol honestly; see [I-D.trammell-stackevo-explicit-coop] for
more.

7.5. No unnecessary restrictions on the superstrate

Beyond those restrictions deemed necessary as common features of any
secure, responsible transport protocol (see Section 6.7), SPUD must
impose only minimal restrictions on the transport protocols it
encapsulates. However, to serve as a substrate, it is necessary to
factor out the information that middleboxes commonly rely on and
endpoints are commonly willing to expose. This information should be
included in SPUD, and might itself impose additional restrictions to
the superstrate.

7.6. Minimal additional start-up latency

SPUD should not introduce additional start-up latency for
superstrates.

7.7. Minimal header overhead

To avoid reducing network performance, the information and coding
used in SPUD should be designed to use the minimum necessary amount
of additional space in encapsulation headers.

7.8. Minimal non-productive traffic

SPUD should minimize additional non-productive traffic (e.g.
keepalives), and should provide mechanisms to allow its superstrates
to minimize their reliance on non-productive traffic.

7.9. **Endpoint control over reverse-path middlebox signaling**

   In some cases, a middlebox may need to send a packet directly in
   response to a sending endpoint, e.g. to signal an error condition.
   In this case, the direct return packet generated by the middlebox
   uses the reversed end-to-end 5-tuple in order to receive equivalent
   NAT treatment, though the reverse path might not be the same as the
   forward path. Endpoints have control over this feature: A SPUD-aware
   middlebox must not emit a direct return packet unless it is in direct
   response to a packet from a receiving endpoint, and must not forward
   a packet for which it has sent a direct return packet.

7.10. **Reliability, Fragmentation, MTU, and Duplication**

   As any information provided by SPUD is anyway opportunistic, SPUD
   need not provide reliable signaling for the information associated
   with a tube. Signals must be idempotent; all middleboxes and
   endpoints must gracefully handle receiving duplicate signal
   information. SPUD must continue working in the presence of IPv4
   fragmentation on path, but in order to reduce the impact of requiring
   fragments reassembly at middleboxes for signals to be intelligible,
   endpoints using SPUD should attempt to fit all signals into single
   MTU-sized packets.

   Given the importance of good path MTU information to SPUD's own
   signaling, SPUD should implement packetization layer path MTU
   discovery [RFC4821].

   Any facilities requiring more than an MTU's worth of data in a single
   signal should use an out-of-band method which does provide
   reliability - this method may be an existing transport or
   superstrate/SPUD combination, or a "minimal transport" defined by
   SPUD for its own use.

7.11. **Interoperability with non-encapsulated superstrates**

   It is presumed that "superstrate X with SPUD" is a distinct entity on
   the wire from "superstrate X". The APIs the superstrate presents to
   the application should be equivalent, and the two wire protocols
   should be freely transcodeable between each other, with the caveat
   that the variant without SPUD would not necessarily support features
   enabling communication with the path. However, there is no
   requirement that the headers the superstrate uses be the same in the
   SPUD and non-SPUD variants. Headers that the superstrate chooses
   always to expose to the path can therefore be encoded in the SPUD
   layer but not appear in an upper-layer header.

8. **Open questions and discussion**

   The preceding requirements reflect the present best understanding of
   the authors of the functional and technical requirements on an
   encapsulation-based protocol for common middlebox-endpoint
   cooperation for superstrates. There remain a few large open
   questions and points for discussion, detailed in the subsections
   below.

8.1. **Property binding**

   Related to identifier scope is the scope of properties bound to SPUD
   packets by endpoints. SPUD may support both per-tube properties as
   well as per-packet properties. Properties signaled per packet reduce
   state requirements at middleboxes, but also increase per-packet
   overhead. Small signal size (in bits of entropy) and encoding
   efficiency (in bits on the wire) is therefore more important for per-
   packet signaling that per-tube signaling.

   It is likely that both types of property binding are useful, but the
   selection of which properties to bind how must be undertaken
   carefully. It is also possible that SPUD will provide a very limited
   set of per-packet signals (such as ECN) using flags in the SPUD
   header, and require all more complicated properties to be bound per-
   tube.

8.2. **Tradeoffs in integrity protection**

   In order to protect the integrity of information carried by SPUD
   against forging by malicious devices along the path, it would be
   necessary to be able to authenticate the originator of that
   information. We presume that the authentication of endpoints is a
   generally desirable property, and to be handled by the superstrate;
   in this case, SPUD may be able borrow that authentication to protect
   the integrity of endpoint-originated information.

   However, in the Internet, it is not in the general case possible for
   the endpoint to authenticate every middlebox that might see packets
   it sends and receives. In this case information produced by
   middleboxes may enjoy less integrity protection than that produced by
   endpoints. In addition, endpoint authentication of middleboxes and
   vice-versa may be better conducted out-of- band (treating the
   middlebox as an endpoint for the authentication protocol) than in-
   band (treating the middlebox as a participant in a 3+ party
   communication).

8.3. Piggybacked, interleaved, and reflected signaling

   The requirements in Section 5.2 and Section 5.3 are best met by in-
   band signaling: packets carrying the same 6-tuple as packets
   containing superstrate headers and payload.

   Path-to-endpoint signaling can be piggybacked and/or interleaved
   (where SPUD and the superstrate each have their own packets).

   In-band signaling has the advantage that it does not require
   foreknowledge of the identity and addresses of devices along the path
   by endpoints and vice versa, but does add complexity to the signaling
   protocol. Piggybacked signaling uses some number of bits in each
   packet generated by the overlying transport. It requires either
   reducing the MTU available to the encapsulated transport and/or
   opportunistically using "headroom" as it is available: bits between
   the network-layer MTU and the bits actually used by the transport.
   For use cases that accumulate information from devices on path in the
   SPUD header, piggybacked signaling also requires a mechanism for
   endpoints to create "scratch space" for potential use of the on-path
   devices. In contrast, interleaved signaling uses signaling packets
   on the same 5-tuple and tube ID, which don't carry any superstrate
   data. These interleaved packets can also contain scratch space for
   on-path device use. This reduces complexity and sidesteps MTU
   problems, at the cost of sending more packets per flow.

8.4. Continuum of trust among endpoints and middleboxes

   There are different security considerations for different security
   contexts. The end-to-end context is one; anything that only needs to
   be seen by the path shouldn't be exposed in SPUD, but rather by the
   superstrate. There are multiple different types of end-to-middle
   context based on levels of trust between end and middle - is the
   middlebox on the same network as the endpoint, under control of the
   same owner? Is there some contract between the application user and
   the middlebox operator? SPUD should support different levels of
   trust than the default ("untrusted, but presumed honest due to
   limitations on the signaling vocabulary") and fully-authenticated;
   how these points along the continuum are to be implemented and how
   they relate to each other needs to be explored further.

8.5. Discovery and capability exposure

   There are two open issues in discovery and capability exposure.
   First, endpoints might want to be able to discover SPUD-aware
   middleboxes along the path, and to discover which parts of the
   vocabulary that can be spoken by the endpoints are supported by those
   middleboxes as well as the other communication endpoint, and vice

versa. Second, SPUD coult assist endpoints in discovering and
negotiate which superstrates are available for a given interaction,
though it is explicitly not a goal of SPUD to expose information
about the details of the superstrate to middleboxes.

8.6. Hard state vs. soft state

The initial thinking on signaling envisions "hard state" in
middleboxes that is established when the middlebox observes the start
of a SPUD tube and is torn down when the middlebox observes the end
(stop) of a SPUD tube. Such state can be abandoned as a result of
network topology changes (e.g., routing update in response to link or
node failure). An alternative is a "soft state" approach that
requires periodic refresh of state in middleboxes, but cleanly times
out and discards abandoned state. SPUD has the opportunity to use
different timeouts than the defaults that are required for current
NAT and firewall pinhole maintenance. Of course, applications will
still have to detect non-SPUD middleboxes that use shorter timers.

8.7. Tube vs. superstrate association lifetime

The requirements as presently defined use tube start and stop
signaling for two things: (1) setting up and tearing down state along
the path, and (2) signaling superstrate such as association startup,
acceptance, and teardown, which may have security implications.
These may require separate signaling. Specifically, if tube start
acknowledgment is to be used to provide explicit guarantees to the
path about the acceptability of a tube to a remote endpoint, it
cannot be a completely unreliable signal. Second, the lifetime of a
tube may be much shorter than the lifetime of a superstrate
association, and the creation of a new tube over an existing
association may need to be treated differently by endpoints and path
devices than a tube creation coincident with an association creation.

8.8. SPUD Support Discovery

If SPUD is not usable on a path to an endpoint, a SPUD sender needs
to be able to fall back to some other approach to achieve the goals
of the superstrate; a SPUD endpoint must be able to easily determine
whether a remote endpoint with which it wants to communicate using
SPUD as a substrate can support SPUD, and whether path to the remote
endpoint as well as the return path from the remote endpoint will
pass SPUD packets.

It is not clear whether this is a requirement of SPUD, or a
requirement of the superstrate / application over SPUD.

9. Security Considerations

The security-relevant requirements for SPUD are outlined in
Section 6. In addition, security-relevant open issues are discussed
in Section 8.2 and Section 8.4. These will be further addressed in
protocol definition work following from these requirements.

10. IANA Considerations

This document has no actions for IANA.

11. Contributors

In addition to the editors, this document is the work of David Black,
Ken Calvert, Ted Hardie, Joe Hildebrand, Jana Iyengar, and Eric
Rescorla.

12. Acknowledgments

Thanks to Ozgu Alay, Roland Bless, Cameron Byrne, Toerless Eckert,
Gorry Fairhurst, Daniel Kahn Gillmor, Tom Herbert, Christian Huitema,
Iain Learmonth, Diego Lopez, and Matteo Varvelli for feedback and
comments on these requirements, as well as to the participants at the
SPUD BoF at IETF 92 meeting in Dallas and the IAB SEMI workshop in
Zurich for the discussions leading to this work.

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

[RFC0792]   Postel, J., "Internet Control Message Protocol", STD 5,
            RFC 792, DOI 10.17487/RFC0792, September 1981,
            <http://www.rfc-editor.org/info/rfc792>.

[RFC2827]   Ferguson, P. and D. Senie, "Network Ingress Filtering:
            Defeating Denial of Service Attacks which employ IP Source
            Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827,
            May 2000, <http://www.rfc-editor.org/info/rfc2827>.

[RFC3234]   Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and
            Issues", RFC 3234, DOI 10.17487/RFC3234, February 2002,
            <http://www.rfc-editor.org/info/rfc3234>.

   [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally
              Unique IDentifier (UUID) URN Namespace", RFC 4122,
              DOI 10.17487/RFC4122, July 2005,
              <http://www.rfc-editor.org/info/rfc4122>.

   [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>.

   [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
              IANA Considerations Section in RFCs", BCP 26, RFC 5226,
              DOI 10.17487/RFC5226, May 2008,
              <http://www.rfc-editor.org/info/rfc5226>.

   [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
              Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,
              January 2012, <http://www.rfc-editor.org/info/rfc6347>.

   [RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black,
              "Encapsulating MPLS in UDP", RFC 7510,
              DOI 10.17487/RFC7510, April 2015,
              <http://www.rfc-editor.org/info/rfc7510>.

   [RFC7663] Trammell, B., Ed. and M. Kuehlewind, Ed., "Report from the
              IAB Workshop on Stack Evolution in a Middlebox Internet
              (SEMI)", RFC 7663, DOI 10.17487/RFC7663, October 2015,
              <http://www.rfc-editor.org/info/rfc7663>.

   [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.kuehlewind-spud-use-cases]
              Kuehlewind, M. and B. Trammell, "SPUD Use Cases", draft-
              kuehlewind-spud-use-cases-00 (work in progress), July
              2015.

   [I-D.huitema-tls-dtls-as-subtransport]
              Huitema, C., Rescorla, E., and J. Jana, "DTLS as
              Subtransport protocol", draft-huitema-tls-dtls-as-
              subtransport-00 (work in progress), March 2015.

   [I-D.trammell-stackevo-explicit-coop]
              Trammell, B., "Architectural Considerations for Transport
              Evolution with Explicit Path Cooperation", draft-trammell-
              stackevo-explicit-coop-00 (work in progress), September
              2015.

Authors' Addresses

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

   Email: ietf@trammell.ch

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

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