Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: August 20, 2012

T. Ayar
B. Rathke
L. Budzisz
A. Wolisz
Technical University Berlin
February 17, 2012

A Transparent Performance Enhancing Proxy Architecture To Enable TCP over Multiple Paths for Single-Homed Hosts
draft-ayar-transparent-sca-proxy-00

Abstract

This draft complements the work of MPTCP by defining a TCP Splitter/ Combiner Architecture (SCA) that enables non-MPTCP-capable single-homed hosts to benefit from the multiple paths within Internet by means of performance enhancing proxies (PEPs) placed in the access networks.

SCA Proxies (SCAPs) make use of multiple paths in a way which is completely transparent to end-hosts. Since the existence of the SCAPs is shielded from the TCP end-points, they can be deployed in the Internet as well as on the end-systems.

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 August 20, 2012.

Table of Contents

1. Introduction

Despite the fact that, the number of end-systems with multiple interfaces increases every day, usually only one interface is used to connect to the Internet, as for the single-interface hosts. This is motivated, among others, by the energy consumption. That is, the use of multiple interfaces drastically increases the power consumption and thus decreases the battery lifetime. When only one IP address is assigned to the end-system, the end-system is single-addressed.

Single addressed end-hosts may not benefit from the MPTCP [2] [3] [4] even if they are MPTCP-capable. MPTCP uses IP address pairs of the end-hosts to create the subflows. When two MPTCP-capable single-addressed end-systems transfer data via MPTCP, they will use regular TCP [5] like the non-MPTCP-capable hosts.

Moreover, the fact that an end-system is MPTCP-capable does not mean that MPTCP MUST be used by TCP connections. MPTCP MAY be disabled with TCP_MULTIPATH_ENABLE socket option on MPTCP-capable hosts [6].

Nevertheless, single-addressed end-hosts may benefit from multiple paths within the Internet by means of proxies located within the access networks. Such a proxy is described in this document and called further SCA Proxy (SCAP). It detects the TCP connection and then splits and shapes TCP traffic over multiple paths. If the access network is connected to the Internet via multiple gateway (GW) links, then TCP traffic may be distributed within the access network so that the packets will leave the network from different GW links. Thus, the bandwidth aggregation of the links may be achieved. If an access network is connected to the Internet via only one GW link, the SCAP may distribute TCP traffic to multiple paths within the access network and then another SCAP on the path may combine the packets before they leave the access network.

This document describes a splitter/combiner architecture (SCA) that may be used to develop transparent proxies for TCP over multiple paths solutions. SCA is a thin layer on top of IP which captures all the TCP packets. SCA distributes data/ACK packets of a TCP connection to the multiple paths and/or combines the distributed data/ACK packets of a TCP connection from multiple paths.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

1.2. Terminology

Single-addressed end-system: An end-system, either with single or multiple interfaces, which is assigned only one IP address to connect to the Internet.

Network Device: An access point (AP) or router located in between a TCP sender and receiver.

SCA Proxy (SCAP): A running instance of the SCA.

SCAP Device: A network device on which an instance of SCA runs.

Pipe: A network path between a network interface of a SCAP device and a network interface of another SCAP device. Pipe is uniquely identified by the IP address pair of two SCAP device interfaces.

1.3. Reference Scenario

Figure 1 shows a reference scenario for two SCAP devices in different access networks. Host A and B are single-addressed non-MPTCP-capable hosts. They are connected to the Internet via network devices which have multiple (i.e., N) GW links to the Internet. The multiple interfaces of the network devices have IP addresses A-1..N and B-1..N.

When there is a TCP connection between an application on Host A and an application on Host B, TCP packets will follow a single path (e.g., via A-1<->Pipe-1<->B-1). If the GW links have low capacities (e.g., T1 links or DSL lines), then they constitute the bottleneck for the TCP connection.

As shown in Figure 1, when network devices are used as SCAP devices they may use multiple GW links to aggregate the capacity of the links for the TCP connection:

- SCAP-1 splits TCP data packets from Host A to Host B via its interfaces A-1 to A-N. Similarly, SCAP-2 splits TCP data packets from Host B to Host A via its interfaces B-1 to B-N.

- On the other side, SCAP-1 combines the packets split by SCAP-2. That is, it buffers and reorders the packets before they reach to the Host A. Similarly, SCAP-2 combines the packets split by SCAP-1.

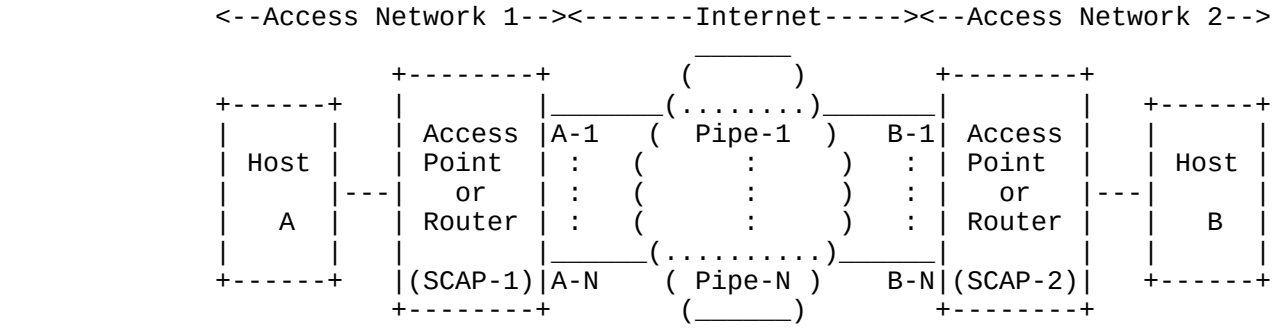

Figure 1: SCAPs on Multiple Access Networks

2. SCA Design

In addition to increasing the TCP throughput, SCA aims at high deployment and adoption. To achieve that, the existence of the SCA MUST be shielded locally (i.e., from the applications and TCP on the TCP end-points) as well as on end-to-end (i.e., end points SHOULD NOT need to know which network devices are SCAP devices). Thus, SCA is designed as protocol-stack and end-host transparent.

2.1. SCA Protocol-Stack Transparency

In order to provide protocol-stack transparency, SCA MUST be located underneath the TCP. All the TCP multiple path related issues MUST be handled in the SCA which MAY be implemented as a thin layer.

SCA SHOULD be located on top of the IP on the network devices. Alternatively, SCA MAY be located between the IP and data link layer. In both cases, all the TCP packets MUST pass through the SCA.

In order to recognize TCP connection establishment/release requests and TCP data/ACK packets, SCA MUST access and read the content of the TCP header. Thus, SCA SHOULD NOT be used on network devices if TCP headers can not be read. SCA SHOULD NOT access and read TCP payload.

Network devices MAY access TCP headers to work as middleboxes (i.e., performance-enhancing proxies (PEPs), firewalls, or NATs) [2]. As a kind of middlebox, SCAP devices function as PEPs that split and combine TCP traffic while preserving TCP end-to-end semantics (Figure 2).

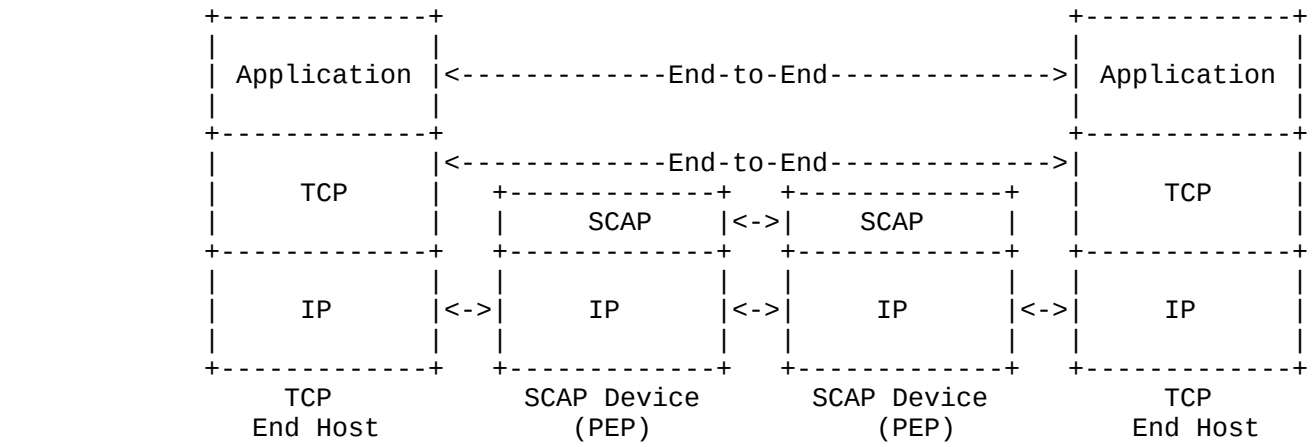

Figure 2: SCAPs as PEPs In Between TCP End Hosts

2.2. SCA End-Point Transparency

SCAPs MUST work without any support from TCP end hosts. Neither TCP sender nor TCP receiver MUST be aware of the presence of any SCAPs. Thus, TCP end-hosts SHALL NOT be configured to find SCAPs. Instead, SCAPs SHOULD find other SCAPs to collaborate with.

3. SCA Components

SCA includes the following components to solve TCP problems [7] when multiple pipes are used:

- Packet Classifier that captures TCP packets and classifies them based on the header information.
- Connection Handler that reacts to new TCP connections by creating records of the new TCP connection and discarding connection information in case of a TCP connection release.
- Multiple Pipes Adapter that provides a common interface to use pipes in the network layer.
- Data/ACK Processors that process the data/ACK packets of the TCP connections.
- Signaling Unit that processes the SCA signaling packets.
- Configuration and Management Unit that is used to configure parameters used by the components.

3.1. Packet Classifier

Packet Classifier is responsible for looking at the TCP header to determine the packet type. An admission control policy MAY be applied to packets (Section 3.6) before accepting them into the SCAP. The packets that are not admitted MUST be passed to the lower/upper layer without SCAP processing.

SCAP-accepted TCP packets MUST be passed to the related SCA component:

- SCA signaling packets are used for detecting other SCAPs and exchanging control information among the SCAPs. All the signaling packets MUST be handled by the Signaling Unit.
- TCP connection establishment packets (i.e., packets with the SYN flag set) MUST be passed to the Connection Handler to construct connection record entries.
- TCP connection release packets MUST be passed to Connection Handler to remove entries of the connection records.
- Data/ACK packets MUST be passed to the Data/ACK Processor.

3.2. Connection Handler

Connection Handler manages the records of the TCP connections. TCP connection establishment procedure is followed to detect the TCP connections. TCP connection requests are detected by means of TCP segments with SYN flag.

TCP connection records MUST contain a unique connection identifier (tcp_conn_id) that is generated based on the TCP sender and receiver end-points (i.e., sender and receiver IP addresses and port numbers).

When TCP connection release packets (i.e., packets with the FIN or RST flag set) are received, the TCP connection release procedure is followed. When the connection tear-down is complete, all the connection related records MUST be deleted.

3.3. Multiple Pipes Adapter

MPTCP uses IP addresses of the end-hosts in the subflow end-points. Similarly, SCAPs SHOULD use IP address pairs of SCAP devices as pipes. In that case, the peer MUST be informed about the used addresses, as MPTCP informs its peer by means of ADD_ADDR option.

TCP packets MAY be sent over these pipes to the SCAP peer by means of:

- IP source routing [8]:

  TCP packets may be requested to follow a specific path. IP Loose/ Strict Source and Record Route option MAY be used to specify the path they must follow.

- IP-in-IP encapsulation [9]:

  The source and destination SCAP IP addresses may be specified in the outer IP header to create tunnels between the SCAP peers.

- IP-in-IP tunneling [10]:

  SCAPs may use special headers to exchange signaling information. Tunnel Headers in the IP-in-IP tunneling may be used to carry SCA signaling information.

Since different methods may be used to send/receive data via SCA pipes, Multiple Pipes Adapter is defined as a component that interacts via specified APIs with other components of the SCA and handles interaction with IP.

Multiple Pipes Adapter interacts with IP to access and use the available pipes. Other SCA components use the Multiple Pipes Adapter APIs to get information about the pipes that may be used and to send data via them.

Multiple Pipes Adapter provides following APIs to the SCA components:

1. PIPE-ID-LIST find_pipes(tcp_conn_id, destination_address)

   is used to find pipes that may be used to send TCP data packets to the destination_address and returns the list of them, if any.

   tcp_conn_id is used to identify the connection that needs the pipes to be assigned. Multiple Pipes Adapter may access information about the connection (e.g., source/destination TCP end-point) by using the tcp_conn_id. The API also associates the list of pipes found with the connection.

   The PIPE-ID-LIST contains pipe_ids which are assigned by the Multiple Pipes Adapter to the available pipes (i.e., each pipe has a unique pipe_id).

Pipe implementation is shielded from the other components by means of pipe_ids: other components only know pipe_ids and Multiple Pipe Adapter handles pipes.

2. PIPE-PROPERTIES get_pipe_properties(tcp_conn_id, pipe_id)

   is used to probe for characteristics of a pipe.

   PIPE-PROPERTIES includes the required pipe parameters by the SCA instance (e.g., bandwidth, delay, utilization, packet loss rate, etc.).

   In order to get pipe parameters, following methods may be used [7]: packet-pair based estimation of the pipe delays and capacities, tracking the TCP timestamps [11], SACK [12] or D-SACK [13] to estimate packet arrivals at the receiver, or probing pipe capacities with TCP-like cwnd increase and decrease [3].

   In addition, the local MIB information (e.g., [14] [15] [16] [17]) may be used to set the initial values just after the pipes are found by Multiple Pipes Adapter.

   If tcp_conn_id is NULL, then the total pipe characteristics are returned. Otherwise, tcp_conn_id is used to get pipe characteristics relevant to a TCP connection. For example, the bandwidth used by a TCP connection may be asked as well as the total bandwidth of the pipe.

3. PIPE-ID-LIST employ_pipes(tcp_conn_id, pipe_id_list)

   is used by Multiple Pipes Adapter to associate pipes which become available after the connection establishment. For example, an interface becomes active or a host associates with an AP/router after the pipes were assigned to the TCP connection by use of the find_pipes() API.

   Pipes in the pipe_id_list are associated with the connection with tcp_conn_id.

4. PIPE-ID-LIST withdraw_pipes(tcp_conn_id, pipe_id_list)

   is used when a connection terminates or connection doesn’t need some pre-assigned pipes.

   Pipes in the pipe_id_list are disassociated from the connection with tcp_conn_id.

It MUST be called after the connection termination and MAY be called during the connection to cancel the use of some pipes which are previously assigned to the connection.

In addition, Multiple Pipes Adapter may use this API to disassociate some pipes which become unavailable during the connection. For example, an interface becomes inactive or a host looses its connection to an AP/router.

5. PIPE-ID-LIST select_pipes(tcp_conn_id, pipe_id_list)

   is used to select a subset from a set of pipes based on a pipe selection policy (Section 3.6).

   tcp_conn_id is used to determine the pipe selection policy for the connection. The API uses withdraw_pipes() API to disassociate filtered pipes by the pipe selection policy and returns back the remaining pipes.

6. send_mp_packet(tcp_packet, tcp_conn_id, pipe_id)

   is used to send a packet via one of the multiple pipes.

   Since other SCA components only know the pipe_ids and do not know what the real pipes are, they use this API to send a tcp_packet which belongs to the connection with tcp_conn_id by using the the pipe with pipe_id.

   tcp_conn_id is supplied to enable update for pipe characteristics related with the TCP connections. For example, if a pipe is assigned to more than one TCP connection, what proportion of the pipe is used by the assigned TCP connections.

7. TCP-PACKET check_encapsulation(tcp_packet)

   is used to check whether tcp_packet is encapsulated or not.

   Packets may be sent/received encapsulated. Since other components are shielded from the use of encapsulation, only the Multiple Pipes Adapter knows about the encapsulation.

   send_mp_packet() API encapsulates packets before sending them when encapsulation is necessary.

   When packets are received by the Packet Classifier, it uses check_encapsulation() to get decapsulated packets, if encapsulation is used for the connection. It returns the packet immediately if encapsulation is not used.

3.4. Data/ACK Processor

Data/ACK Processor is responsible for handling data/ACK packets. It shapes TCP traffic and decides on the scheduling of packets to the pipes.

Data/ACK Processor may apply to a TCP data/ACK packet one of the following operations (called 4-D) [7]:

1. duplicate: TCP packets may be (one or more times) duplicated and scheduled to multiple pipes to create robustness against packet losses.

2. delay: TCP packets may be buffered and delivered later, based on a timer or a condition. For example, some packets of a TCP flow may be intentionally delayed to shape the TCP traffic (e.g., to reduce number of out-of-order packet arrivals at the receiver side, or to separate sent packets by a given interval). Another example is to buffer and reorder out-of-order packets before they are passed to the TCP receiver.

3. deliver: TCP packets may be released immediately after their capture, or after the delay or duplicate operation. Delivery may be done locally for the incoming packets or over any available pipe to outgoing packets. For outgoing packets, one of the available pipes must be selected to efficiently utilize the pipes.

4. drop: TCP packets may be dropped by SCAP. For example, assume that a packet is buffered for early retransmission purposes. If the ACK for that packet arrived and the buffered packet is not needed to be kept any more, it may be dropped.

3.5. Signaling Unit

SCAPs on different SCAP devices MAY exchange signaling information by means of the following methods:

1. in-band signaling, by means of TCP options as in MPTCP. The SCAP that wants to send a signaling information will generate an SCA option and add it to the TCP packet. The peer SCAP will use the content of the SCA control option and remove it from the packet.

   The main drawback of the in-band signaling is that its size is limited by the TCP option space allowed. 16-21 bytes of space is left for the SCA options when the most common TCP options are encountered [4].

2. out-of-band signaling that may use signaling channels established between the SCAPs.

   SCAPs may use a well-known port number to exchange signaling packets (e.g, RIP routing processes use UDP port 520 [18] or BGP systems use TCP port 179 [19]).

3. hybrid signaling that may use in-band signaling to carry out-of-band signaling channel end-point information between SCAPs. First a connection between the signaling channel end-points will be established. Then, out-of-band signaling over the connection may be used to exchange the signaling data.

Details of the signaling mechanism are out-of-scope of this document.

3.6. Configuration and Management Unit

Configuration and Management Unit is used to set parameters for the SCA components. The configuration parameters may be defined for each component:

- Packet Classifier may accept packets into the SCAP or reject them based on an admission control policy.

  Based on the TCP end-points, some applications or hosts may not benefit from the SCA:

  * Splitting the traffic may not be necessary for some applications. For example, short-lived flows like web traffic may be carried over only one path instead of multiple paths as they will most likely end within a couple of packets anyway. Thus, they may not be accepted for the SCA processing.

  * SCA may be provided as a service for some set of users. The users may be identified from the IP addresses they use to connect to the Internet. Thus, an admission policy may be applied based on the IP addresses of the TCP end-points.

- Connection Handler may configure the number of acceptable connections based on the current number of connections or the total number of pipes used by the connections.

- Multiple Pipes Adapter may use a pipe selection policy to select a subset of the pipes from the set of available pipes. Using different policies in pipe selection gives additional flexibility to the SCA.

Pipe selection policy may be based on the pipe parameters (e.g., delay, loss rate, throughput, etc.) to decide which pipes will be selected. For example, only the pipes that have close RTT values may be selected or some pipes with high BERs may be excluded.

In addition, pipe selection policy may be based on the TCP end-points. For example, some privileged flows (e.g., flows originated from some IP addresses) may be assigned more pipes and some other flows may be penalized and forced to use only a single pipe.

4. SCA Deployment Scenarios

SCAPs may work in three modes:

1. Standalone SCAP: If there is only one SCAP on the path between the TCP sender and the receiver, SCAP must work alone

2. SCAP Pair: If there are two SCAPs on the path, then they may work as a peer

3. SCAP Chain: If there are multiple SCAPs on the path, then they may be used as cascaded chains

4.1. Standalone SCAP

As shown in Figure 3, SCA may still be used when there is only one SCAP on the path between the TCP end-hosts.

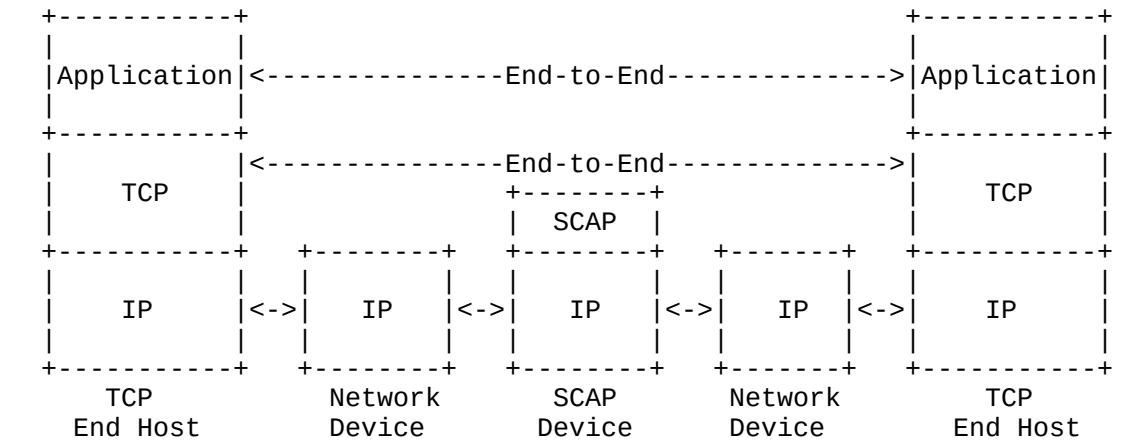

Figure 3: Single SCAP Between TCP End Hosts

If there are multiple pipes from the SCAP to the TCP end-hosts, the SCAP may split and shape TCP traffic over them (e.g., [20]). Since there is only one SCAP, it must work without any feedback via the signaling messages of the peer SCAP. Therefore, SCAP MUST NOT use signaling information during data transfer.

TCP end-hosts MUST NOT notice the existence of the SCAP. The TCP end-hosts will only see the effects of TCP traffic shaping.

4.2. SCAP Peer

As shown in Figure 2, SCAPs MAY work as a peer. The SCAP device peer may be located on the same access network as well as different access networks.

Section 1.3 discusses a scenario of SCAP devices located on different access networks.

Figure 4 shows two SCAP devices within the same access network. A TCP connection is established between an application on the single-homed Host A to an application on single-homed Host B.

In the data flow from Host A to Host B, SCAP-1 may split TCP packets via Pipe-1 and Pipe-2. This distribution may be used for load balancing purposes within the access network. In addition, if Pipe-1 and Pipe-2 are the bottleneck, then bandwidth aggregation of the pipes may be achieved. SCAP-2 combines the distributed packets, re-orders them and sends them in-order to the Internet. Similarly, for the data flow from Host B to Host A, SCAP-2 may split packets over Pipe-1 and Pipe-2 while SCAP-1 may combine them.

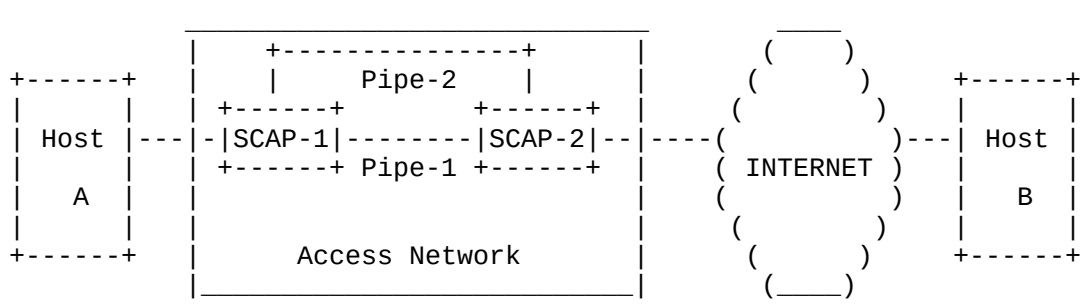

Figure 4: SCAPs on The Same Access Network

4.3. SCAP Chain

As shown in Figure 5, more than two SCAPs may be available on the path between the TCP end-hosts. In that case, SCAP peers constitute a SCAP chain. A SCAP distributes TCP packets over multiple pipes, and its peer combines them. This split/combine process is then repeated between each SCAP pair.

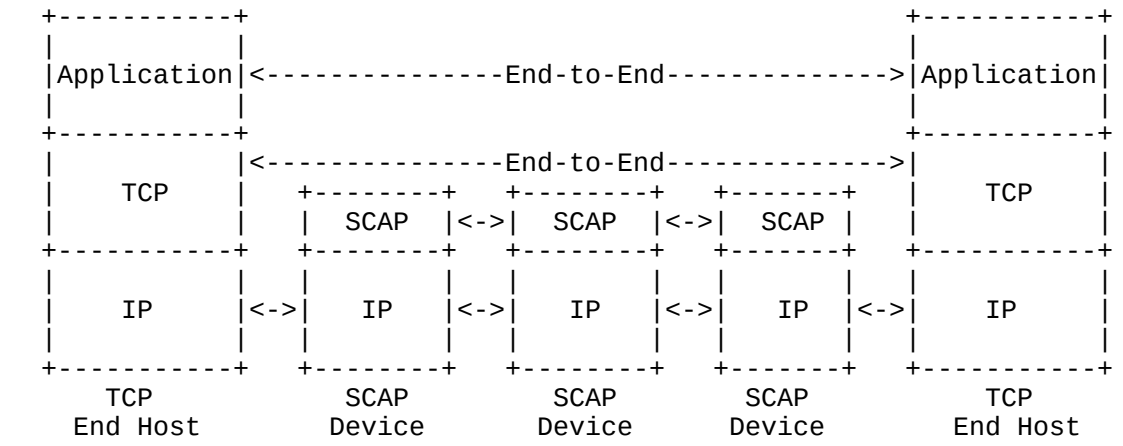

Figure 5: A SCAP Chain Between TCP End Hosts

5. SCA Extensions

Since SCA is a thin layer underneath the TCP, it MAY also be placed on multi homed TCP end-hosts. SCA SHOULD be located in between the TCP and IP when it is used on the TCP end-hosts. Alternatively, SCA MAY be located between the IP and data link layer.

Figure 6 shows a SCAP peer on TCP end-hosts. SCA may be also used in the standalone setting (i.e., only on one TCP end-host).

Neither TCP nor IP SHOULD perceive the intervention of the SCAP:

- TCP sends outgoing TCP packets to its lower layer entity.
- IP passes incoming TCP packets to its upper layer entity.

In both cases, SCAP captures packets and processes them without TCP or IP having knowledge about that.

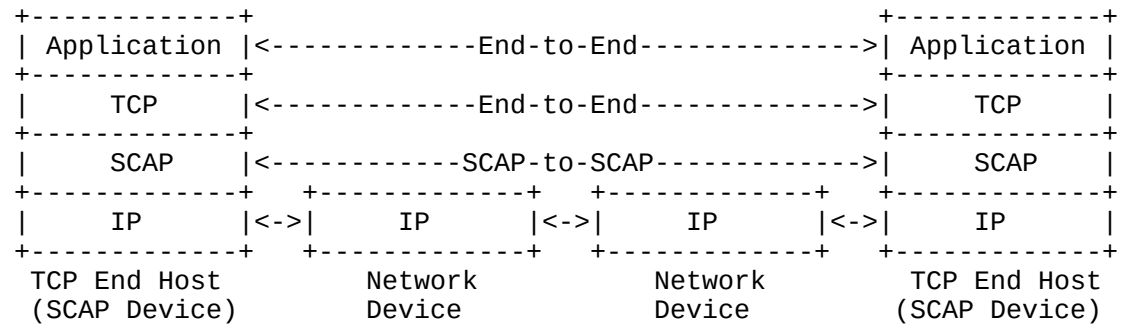

Figure 6: SCAPs on TCP End-Hosts

Finally, in the most general case, SCAPs on TCP end-hosts MAY collaborate with the SCAPs on network devices as shown in Figure 7.

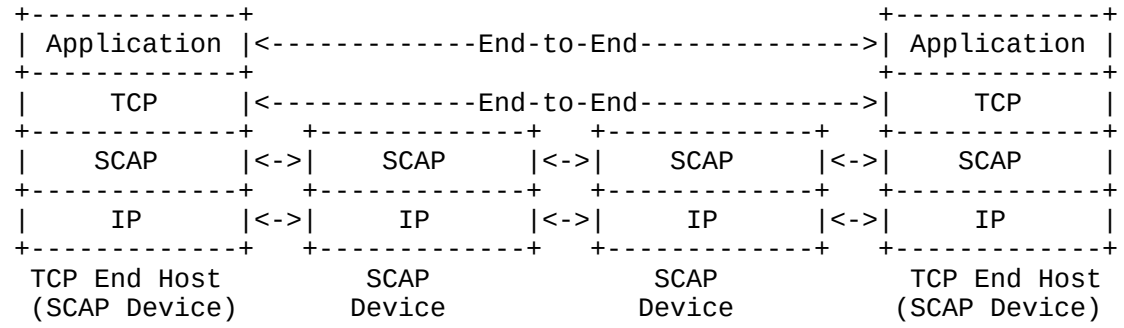

Figure 7: SCAPs as End-Hosts and Routers

6. IANA Considerations

This draft includes no IANA considerations.

7. Security Considerations

Will be added in a later version of this document.

8. References

8.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

[2] Ford, A., Raiciu, C., Handley, M., Barre, S., and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", RFC 6182, March 2011.

[3] Raiciu, C., Handley, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, October 2011.

[4] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", draft-ietf-mptcp-multiaddressed-06 (work in progress), January 2012.

[5] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[6] Scharf, M. and A. Ford, "MPTCP Application Interface Considerations", draft-ietf-mptcp-api-03 (work in progress), November 2011.

[7] Ayar, T., Rathke, B., Budzisz, L., and A. Wolisz, "A Splitter/ Combiner Architecture for TCP over Multiple Paths", TKN Technical Report Series TKN-12-001, February 2012, <http:// www.tkn.tu-berlin.de/menue/publications/ tkn_technical_report_series/>.

[8] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[9] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.

[10] Simpson, W., "IP in IP Tunneling", RFC 1853, October 1995.

[11] Jacobson, V., Braden, B., and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

[12] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, October 1996.

[13] Floyd, S., Mahdavi, J., Mathis, M., and M. Podolsky, "An

Extension to the Selective Acknowledgement (SACK) Option for TCP", RFC 2883, July 2000.

[14] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.

[15] Raghunarayan, R., "Management Information Base for the Transmission Control Protocol (TCP)", RFC 4022, March 2005.

[16] Routhier, S., "Management Information Base for the Internet Protocol (IP)", RFC 4293, April 2006.

[17] Mathis, M., Heffner, J., and R. Raghunarayan, "TCP Extended Statistics MIB", RFC 4898, May 2007.

[18] Malkin, G., "RIP Version 2", STD 56, RFC 2453, November 1998.

[19] Rekhter, Y. and T. Li, "A Border Gateway Protocol 4 (BGP-4)", RFC 1771, March 1995.

[20] Ayar, T., Rathke, B., Budzisz, L., and A. Wolisz, "TCP over Multiple Paths Revisited: Towards Transparent Proxy Solutions", Accepted by The IEEE International Conference on Communications 2012 (IEEE ICC'12), June 2012.

Authors' Addresses

Tacettin Ayar
Technical University Berlin
Telecommunication Networks Group,
Sekr. FT5, Einsteinufer 25,
Berlin 10587
Germany

Phone: +49 30 314 28225
Email: ayar@tkn.tu-berlin.de

Berthold Rathke
Technical University Berlin
Telecommunication Networks Group,
Sekr. FT5, Einsteinufer 25,
Berlin,   10587
Germany

Phone: +49 30 314 23832
Email: rathke@tkn.tu-berlin.de

Lukasz Budzisz
Technical University Berlin
Telecommunication Networks Group,
Sekr. FT5, Einsteinufer 25,
Berlin,   10587
Germany

Phone: +49 30 314 23836
Email: budzisz@tkn.tu-berlin.de

Adam Wolisz
Technical University Berlin
Telecommunication Networks Group,
Sekr. FT5, Einsteinufer 25,
Berlin,   10587
Germany

Phone: +49 30 314 22911
Email: awo@ieee.org