loops
Internet-Draft
Intended status: Standards Track
Expires: 14 December 2020

C. Bormann
Universitaet Bremen TZI

12 June 2020

Embedding LOOPS in Geneve
draft-bormann-loops-geneve-binding-01

Abstract

LOOPS (Local Optimizations on Path Segments) aims to provide local in-network loss recovery. It can be used with tunneling protocols to efficiently recover lost packets on a single segment of an end-to-end path instead of leaving recovery to the end-to-end protocol, traversing the entire path.

[I-D.welzl-loops-gen-info] defines the information to be carried between LOOPS ingress and egress nodes in a generic way, giving a guideline on defining the common elements to embed LOOPS functions in various tunnel protocols. The present document specifies how to embed LOOPS in the overlay tunnel protocol chosen for the initial LOOPS specification, Geneve [I-D.ietf-nvo3-geneve].

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 https://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 14 December 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

LOOPS (Local Optimizations on Path Segments) aims to provide local in-network loss recovery. The LOOPS problems and opportunities draft [I-D.li-tsvwg-loops-problem-opportunities] illustrates some typical scenarios where LOOPS are applicable. One way to use LOOPS is to map it onto a tunnel protocol. The path segment on which LOOPS is applied then is a tunnel, which can be an existing one or created on purpose.

LOOPS allows the packet loss recovery to be performed over specific segments instead of end-to-end, enabling faster and more reliable data delivery. [I-D.welzl-loops-gen-info] defines the information to be carried between LOOPS ingress and egress nodes in a generic way, giving a guideline on defining the common elements to embed LOOPS functions in various tunnel protocols.

Geneve [I-D.ietf-nvo3-geneve] is an encapsulation protocol that can be used to create overlay tunnels. It defines an extensible TLV structure to carry so-called "tunnel options". The present document employs this flexibility, specifying how to embed LOOPS in Geneve. This specification covers the format and Geneve-specific procedures only: the actual LOOPS function and procedures are defined in [I-D.welzl-loops-gen-info].

LOOPS has two modes of loss recovery, retransmission and forward error correction (FEC). The current version of the present document covers retransmission only.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document makes use of the terminology defined in [I-D.welzl-loops-gen-info].

3. Geneve LOOPS Frame Format

Figure 1 shows the format of the Geneve Header and a single Geneve Option, as defined in [I-D.ietf-nvo3-geneve]. Geneve LOOPS defines a new Option class called LOOPS to carry LOOPS forward and backward information.

Geneve Header and Option:

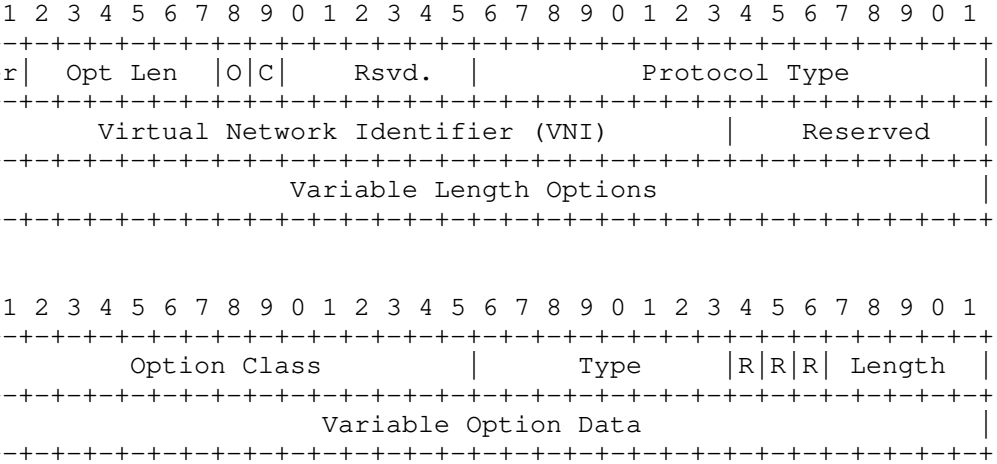


Figure 1: Geneve Header and Option Format

In the Geneve Option structure, a Geneve LOOPS option uses the following values:

* Option Class: TBD1 for LOOPS (see Section 5).

* Type: Based on the substructure already defined in Geneve, which uses bit 0 (the most significant bit) to indicate a critical option (see see Figure 2), LOOPS defines two type numbers: 0 for LOOPS retransmission mode, and 64 for FEC mode. The present document only addresses messages with LType=0.

TBD: Additional type numbers could be defined, possibly obviating the need for some of the flags in the current option structure.

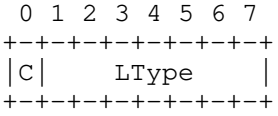


Figure 2: Type Field Format in Geneve LOOPS Option

* C: Critical bit as defined in [I-D.ietf-nvo3-geneve].

* LType: LOOPS Mode.

  - 0: Retransmission mode. In this mode, the LOOPS option format and operations follow this document.

  - 64: FEC mode

  - Further mode values can be assigned in an IANA registry (see Section 5.2).

* Length: Length of Variable Option Data field, expressed in four byte multiples excluding the option header, ranging from 0 to 31. As the option header is another four bytes, the total length of the option in bytes is therefore 4 * (1 + Length), yielding a maximum total length of 128 bytes.

* Variable option data: consists of two parts, Flags and Flag Based Data, as shown in Figure 3.

  - Flags: 16 bits, as described in next subsection. Some of the flags indicate the presence of additional data in the field of Flag Based Data.

- Flag Based Data: This field consists of one or multiple optional data blocks whose presence is indicated by the corresponding flag bits. Any remaining bytes needed to reach a multiple of four bytes are filled with zeroes.

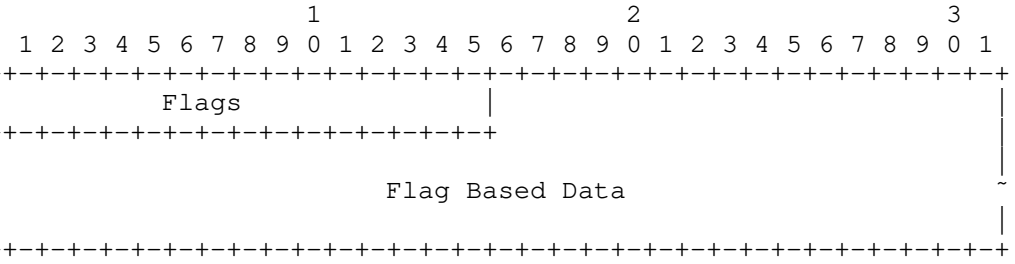


Figure 3: Variable Option Data Format in Geneve LOOPS Option

3.1. Flags and Flag Based Data

Flags for LOOPS Tunnel Options are defined in Figure 4. Some flags cause additional data blocks to occur in the Flags Based Data field. Those additional data blocks are placed in the order of the flags causing them.

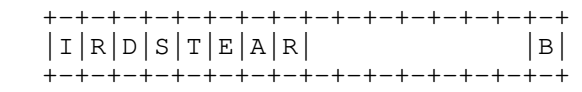


Figure 4: Flags in Variable Option Data in Geneve LOOPS Option

A number of the flag bits are used on their own and do not cause carrying additional data:

* I: Initial Packet Sequence Number (PSN) flag; may be set by the LOOPS ingress to notify the egress about using a new initial PSN.

* R: Initial PSN Received flag; echo of I flag provided by the LOOPS egress.

* D: ACK Desired flag; set by the LOOPS ingress if it wants the egress to generate an acknowledgement immediately upon receiving a particular packet.

These flag bits cause the addition of a single 32-bit number each:

* S: PSN flag; indicates a PSN data block is carried in the Flag Based Data field. It must be set when a packet payload is present. It must not be set if the packet is a pure LOOPS ACK packet, i.e. when no payload is included in the packet.

* T: Timestamp flag. When set, it indicates a Timestamp data block is carried in the Flag Based Data field.
  // TBD: Might want to have "timestamp" and "echo" fields of less or
  // more than 4 bytes.

* E: Echoed Timestamp flag. When set, it indicates an Echoed Timestamp data block is carried in the Flag Based Data field.

* A: ACK number flag. When set, it indicates the presence of a Block 1 ACK information block.

* R: Reception time flag: May only be set if A is set. Indicates that an absolute reception time is given (Format TBD).

Finally, a single flag bit is defined that causes the addition of a variable-length block (therefore this flag is put as the least significant bit of Flags):

* B: Block 2 flag. When set, it indicates the presence a Block 2 ACK information block, with the following format: TBD
  // copy over the structure we have in gen-info.

Acknowledgement information can be sent as a pure ACK packet without payload or piggybacked in a data packet.

4. Security Considerations

The security considerations of [I-D.welzl-loops-gen-info] and [I-D.ietf-nvo3-geneve] apply.

5. IANA Considerations

5.1. Geneve Option Class

IANA is requested to assign a new option class for LOOPS from the "Geneve Option Class" registry.

Option Class	Description
TBD1	LOOPS (Local Optimizations on Path Segments) [RFCthis]

Table 1

5.2. LOOPS Geneve Type Numbers

IANA is requested to create a registry for type numbers ("LType") as used in the TBD1 option class for LOOPS from the "Geneve Option Class" registry, with the following three columns:

Type Number: Integer between 0 and 127

Description: Short Description

Reference: Reference to Specification

The initial contents of the registry is:

Type Number	Description	Reference
0	Retransmission mode	[RFCthis]
64	FEC mode	[RFCthis]

Table 2

(Registry policy TBD, probably Specification Required.)

6. References

6.1. Normative References

[I-D.welzl-loops-gen-info]
Welzl, M. and C. Bormann, "LOOPS Generic Information Set", Work in Progress, Internet-Draft, draft-welzl-loops-gen-info-03, 9 March 2020, <http://www.ietf.org/internet-drafts/draft-welzl-loops-gen-info-03.txt>.

[I-D.ietf-nvo3-geneve]
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", Work in Progress,

Internet-Draft, draft-ietf-nvo3-geneve-16, 7 March 2020, <http://www.ietf.org/internet-drafts/draft-ietf-nvo3-geneve-16.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

6.2. Informative References

[I-D.li-tsvwg-loops-problem-opportunities] Yizhou, L., Zhou, X., Boucadair, M., and J. Wang, "LOOPS (Localized Optimizations on Path Segments) Problem Statement and Opportunities for Network-Assisted Performance Enhancement", Work in Progress, Internet-Draft, draft-li-tsvwg-loops-problem-opportunities-04, 6 January 2020, <http://www.ietf.org/internet-drafts/draft-li-tsvwg-loops-problem-opportunities-04.txt>.

Acknowledgements

Sami Boutros provided some advice on the use of Geneve in this protocol binding.

Author's Address

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org

TSVWG
Internet-Draft
Intended status: Informational
Expires: January 14, 2021

Y. Li
X. Zhou
Huawei
M. Boucadair
Orange
J. Wang
China Telecom
F. Qin
China Mobile

July 13, 2020

LOOPS (Localized Optimizations on Path Segments) Problem Statement and Opportunities for Network-Assisted Performance Enhancement
draft-li-tsvwg-loops-problem-opportunities-06

Abstract

In various network deployments, end to end forwarding paths are partitioned into multiple segments. For example, in some cloud-based WAN communications, stitching multiple overlay tunnels are used for traffic policy enforcement matters such as to optimize traffic distribution or to select paths exposing a lower latency. Likewise, in satellite communications, the communication path is decomposed into two terrestrial segments and a satellite segment. Such long-haul paths are naturally composed of multiple network segments with various encapsulation schemes. Packet loss may show different characteristics on different segments.

Traditional transport protocols (e.g., TCP) respond to packet loss slowly especially in long-haul networks: they either wait for some signal from the receiver to indicate a loss and then retransmit from the sender or rely on sender's timeout which is often quite long. With the increase of end-to-end transport encryption (e.g., QUIC), traditional PEP (performance enhancing proxy) techniques such as TCP splitting are no longer applicable.

LOOPS (Local Optimizations on Path Segments) is a network-assisted performance enhancement over path segment and it aims to provide local in-network recovery to achieve better data delivery by making packet loss recovery faster. In an overlay network scenario, LOOPS can be performed over a variety of the existing, or purposely created, tunnel-based path segments.

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 https://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 January 14, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

1.1. The Problem and Opportunity Overview

Packet loss is ubiquitous in Internet. A reliable transport layer normally employs some end-to-end retransmission mechanisms which also address congestion control [RFC0793] [RFC5681]. The sender either waits for the receiver to send some signals on a packet loss or sets some form of timeout for retransmission. For unreliable transport protocols such as RTP [RFC3550], optional and limited usage of end-to-end retransmission is employed to recover from packet loss [RFC4585] [RFC4588]. End-to-end retransmission to recover lost packets is slow especially when the network is long-haul. For short-lived flows and transactional flows, latency suffers a lot from tail loss.

Tunnels are widely deployed within many networks to achieve various engineering goals, including long-haul WAN interconnection or enterprise wireless access networks. A connection between two endpoints can be decomposed into many connection legs. As such, the corresponding forwarding path can be partitioned into multiple path segments that some of them are using network overlays by means of tunnels. This design serves a number of purposes such as steering the traffic, optimizing egress/ingress link utilization, optimizing traffic performance metrics (such as delay, delay variation, or loss), optimizing resource utilization by invoking resource bonding, provide high-availability, etc.

When a path is partitioned into multiple path segments that are realized typically as overlay tunnels, LOOPS (Local Optimizations on Path Segments) aims to provide in-network recovery over segments to achieve better data delivery by making packet loss recovery faster. In an overlay network scenario, LOOPS can be performed over the existing, or purposely created, overlay tunnel based path segments. Figure 1 show an overall usage scenarios of LOOPS.

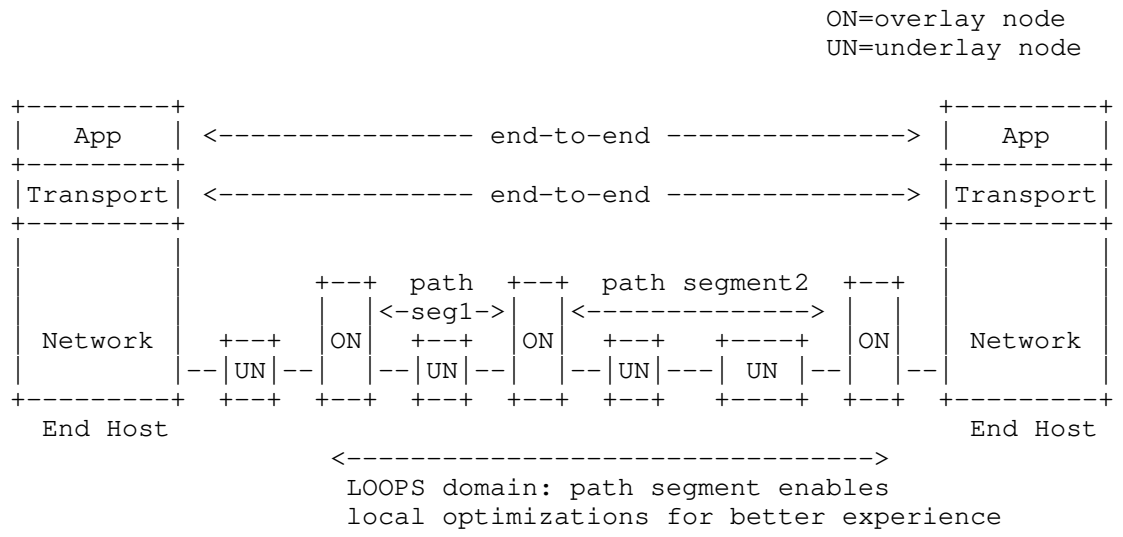


Figure 1: LOOPS Usage Scenario

1.2. Sketching a Work Direction: Rationale & Goals

This document sketches a proposal that is meant to experimentally investigate to what extent a network-assisted approach can contribute to increase the overall perceived quality of experience in specific situations (e.g., Sections 3.5 and 3.6 of [RFC8517]) without requiring access to internal transport primitives. The rationale beneath this approach is that some information (loss detection and segment characteristics, etc.) can be used to trigger local in-network recovery actions which have a faster effect while not impacting the end-to-end congestion control loop.

To that aim, the work is structured into two (2) phased stages:

- Stage 1: Network-assisted optimization. This one assumes that optimizations can be implemented at the network without requiring defining new interaction with the endpoint. Existing tools such as ECN will be used. Loss signal would be converted to CE (congestion experienced) signal to interact with the end-to-end control loop.

- Stage 2: Collaborative networking optimization. This one requires more interaction between the network and an endpoint to implement coordinated and more surgical network-assisted optimizations based on information/instructions shared by an endpoint or sharing locally-visible information with endpoint for better and faster recovery.

The document focuses on the first stage. Effort related to the second stage is out of scope of the initial planned work.

The proposed mechanism is not meant to be applied to all traffic, but only to a subset which is particularly benefits from, and has been selected for the network-assisted optimization service.

Which traffic is selected is deployment-specific and policy-based. For example, techniques for dynamic information about optimization function (e.g., SFC) may be leveraged to unambiguously identify the aggregate of traffic that is eligible to the service. Such identification may be triggered by subscription actions made by customers or be provided by a network provider (e.g., specific applications, during specific events such as during severe DDoS attack or flash crowds events).

Likewise, whether the optimization function is permanently instantiated or on-demand is deployment-specific.

This document does not intend to provide a comprehensive list of target deployment cases. Sample scenarios are described to illustrate some LOOPS potentials. Similar issues and optimizations may be helpful in other deployments such as enhancing the reliability of data transfer when a fleet of drones are used for specific missions (e.g., site inspection, live streaming, and emergency service). Captured data should be reliably transmitted via paths involving radio connections.

It is not required that all segments are LOOPS-aware to benefit from LOOPS advantages.

Section 3 presents the issues and opportunities found in some multiple path segments scenarios. Section 3 describes the impact of packet loss for different traffic. Section 5 describes the LOOPS desired features and their impact on existing network technologies. Section 6 shows the analysis on local retransmission and end-to-end retransmission. Section 7 summarizes LOOPS key elements.

2. Terminology

This document makes use of the following terms:

LOOPS: Local Optimizations on Path Segments. LOOPS includes the local in-network (i.e., non end-to-end) recovery functions and other supporting features such as local measurement, loss detection, and congestion feedback.

LOOPS Node: A node supporting LOOPS functions.

Overlay Node (ON): A node having overlay functions (e.g., overlay protocol encapsulation/decapsulation, header modification, TLV inspection) and LOOPS functions in the LOOPS overlay network usage scenario.

Overlay Tunnel: A tunnel with designated ingress and egress nodes using some network overlay protocol as encapsulation, optionally with a specific traffic type.

Path segment: A LOOPS enabled tunnel-based network subpath. It is used interchangeably with overlay segment in this document when the context wants to emphasize on its overlay encapsulated nature. It is also called segment for simplicity in this document.

Overlay segment: Refers to path segment.

Underlay Node (UN): A node not participating in the overlay network.

3. Usage Scenarios

3.1. Cloud-Internet Overlay Network

CSPs (Cloud Service Providers) are connecting their data centers using the Internet or via self-constructed networks/links. This expands the traditional Internet's infrastructure and, together with the original ISP's infrastructure, forms the Internet underlay.

Automation techniques and NFV (Network Function Virtualization) make it easier to dynamically provision a new virtual node/function as a workload in a cloud for CPU/storage intensive functions. Virtual nodes can be in form of virtual machines or containers hosting the workloads sharing a physical node's infrastructure. With the aid of various mechanisms such as kernel bypassing and Virtual IO, forwarding based on virtual nodes is becoming more and more effective. The interconnection among the purposely positioned virtual nodes and/or the existing nodes with virtualization functions potentially form an overlay infrastructure. It is called the Cloud-Internet Overlay Network (CION) in this document for short.

This architecture scenario makes use of overlay technologies to direct the traffic going through the specific overlay path in order to achieve better service delivery. It purposely creates or selects overlay nodes (ON) from providers. By continuously measuring the delay of path segments and use them as metrics for path selection, when the number of overlay nodes is sufficiently large, there is a high chance that a better path could be found [DOI_10.1109_ICDCS.2016.49] [DOI_10.1145_3038912.3052560]. [DOI_10.1145_3038912.3052560] further shows all cloud providers

experience random loss episodes and random loss accounts for more than 35% of total loss.

Figure 2 shows an example of an overlay path over large geographic distances. An overlay node (ON) is usually a virtual node, though it does not have to be. Three path segments, i.e., ON1-ON2, ON2-ON3, ON3-ON4 are shown. Each segment transmits packets using some form of network overlay protocol encapsulation. ON has the computing and memory resources that can be used for some functions like packet loss detection, network measurement and feedback, packet retransmission and FEC (Forward Error Correction) computation. ONs here are managed by a single administrator though they can be workloads created from different CSPs.

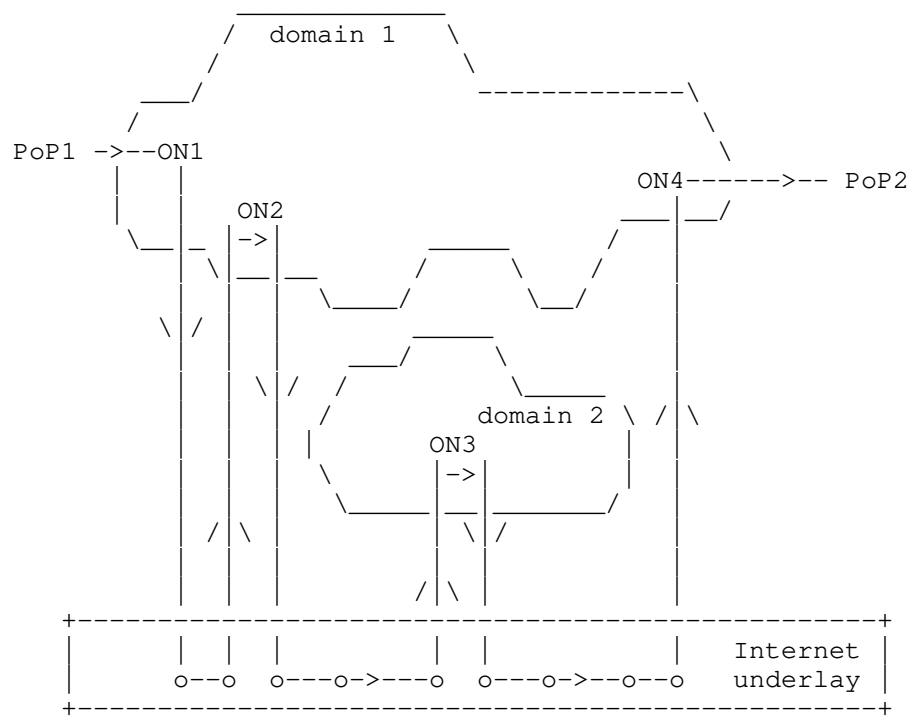


Figure 2: Cloud-Internet Overlay Network (CION)

We tested based on 37 overlay nodes from multiple cloud providers globally. Each pair of the overlay nodes are used as sender and receiver. When the traffic is not intentionally directed to go through any intermediate virtual nodes, we call the path followed by the traffic in the test the default path. When any of the virtual nodes is intentionally used as an intermediate node to forward the

traffic, the path that the traffic takes is called an overlay path. The preliminary experiments showed that the delay of a specifically selected overlay path has lower latency than the one of the default path in 69% of cases at 99% percentile and improvement is 17.5% at 99% percentile when we probe Ping packets every second for a week. The average number of hops for an overlay path is 3.02. More experimental information can be found in [DOI_10.1109_INFCOMW.2019.8845208].

Lower average delay does not necessarily mean less or no packet loss. Different path segments have different packet loss rates. Loss rate is another major factor impacting the user experience, espcially for the short-lived or transactional flows. From some customer requirements, the target loss rate is set in the test to be less than 1% at 99% percentile and 99.9% percentile, respectively. The loss was measured between any two overlay nodes, i.e., any potential path segment. Two thousand Ping packets were sent every 20 seconds between two overlay nodes for 55 hours. This preliminary experiment showed that the packet loss rate satisfaction are only 44.27% and 29.51% at the 99% and 99.9% percentiles, respectively.

As CION naturally consists of multiple overlay segments, LOOPS can leverage this to perform local optimizations on a single hop between two overlay nodes. ("Local" here is a concept relative to end-to-end, it does not mean such optimization is limited to LAN networks.)

3.2. Satellite Communication

Traditionally, satellite communications deploy PEP (performance enhancing proxy [RFC3135]) nodes around the satellite link to enhance end-to-end performance. TCP splitting is a common approach employed by such PEPs, where the TCP connection is split into three: the segment before the satellite hop, the satellite section (uplink, downlink), and the segment behind the satellite hop. This requires heavy interactions with the end-to-end transport protocols, usually without the explicit consent of the end hosts. Unfortunately, this is indistinguishable from a man-in-the-middle attack on TCP. With end-to-end encryption moving under the transport (QUIC), this approach is no longer useful.

Geosynchronous Earth Orbit (GEO) satellites have a one-way delay (up to the satellite and back) on the order of 250 milliseconds. This does not include queueing, coding and other delays in the satellite ground equipment. The Round Trip Time for a TCP or QUIC connection going over a satellite hop in both directions, in the best case, will be on the order of 600 milliseconds. And, it may be considerably longer. RTTs on this order of magnitude have significant performance implications.

Packet loss recovery is an area where splitting the TCP connection into different parts helps. Packets lost on the terrestrial links can be recovered at terrestrial latencies. Packet loss on the satellite link can be recovered more quickly by an optimized satellite protocol between the PEPs and/or link layer FEC than they could be end to end. Again, encryption makes TCP splitting no longer applicable. Enhanced error recovery at the satellite link layer helps for the loss on the satellite link but doesn't help for the terrestrial links. Even when the terrestrial segments are short, any loss must be recovered across the satellite link delay. And, there are cases when a satellite ground station connects to the general Internet with a potentially larger terrestrial segment (e.g., to a correspondent host in another country). Faster recovery over such long terrestrial segments is desirable.

There are two high level classes of solutions for making encrypted transport traffic like QUIC work well over satellite:

- Hooks in the transport protocol which can adapt to large BDPs where both the bandwidth and the latency are large. This would require end to end enhancement.

- Capabilities (such as LOOPS) under the transport protocol to improve performance over specific segments of the path. In particular, separating the terrestrial from the satellite losses. Fixing the terrestrial loss quickly.

This document focuses on the latter.

3.3. Branch Office WAN Connection

Enterprises usually require network connections between the branch offices, or between branch office and cloud data center over geographic distances. With the increasing deployment of vCPE (virtual CPE), services hosted on the CPE are moved to the provider network from the customer site. Such vCPE approach enables some value added service to be provided such as WAN optimization and traffic steering.

Figure 3 shows a branch office access to public cloud via a selected PoP (point of presence) for service access or reaching another branch office via vPC (Virtual Private Cloud) interconnect. vCPE connects to the PoP which can be hundreds of kilometers away via Internet. From vCPE1 to vCPE2, it can consist of three segments, vCPE1-PoP1, PoP1-PoP2 and PoP2-vCPE2. Packet loss can happen on any of them. Segment based in-network recovery can be employed here to improve the WAN connection quality.

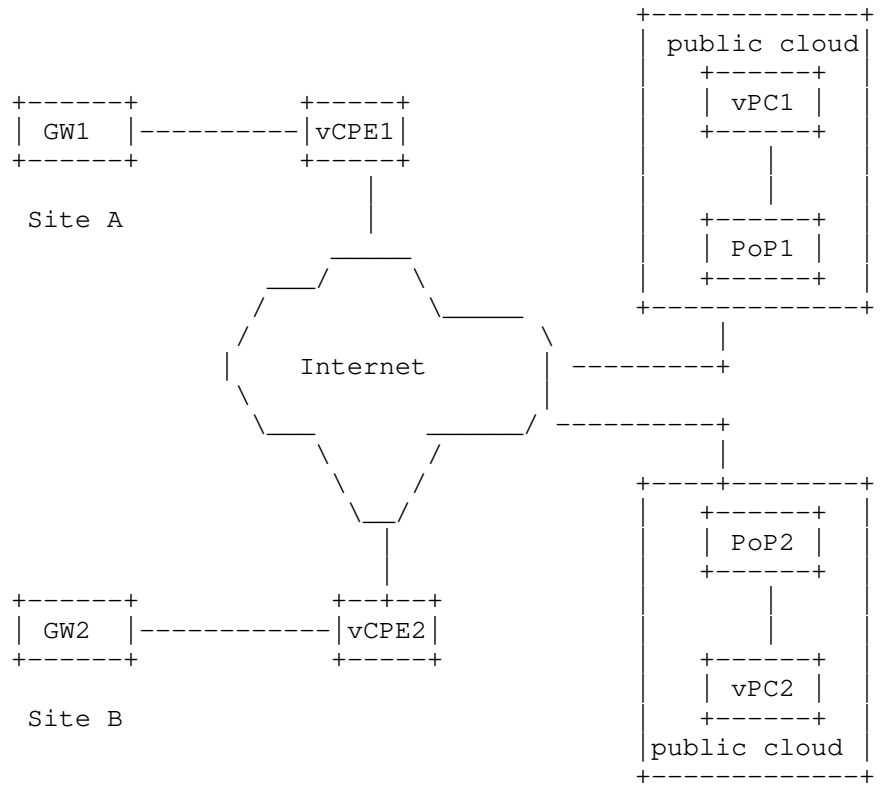


Figure 3: Enterprise Cloud Access

4. Impact of Packet loss

4.1. Tail Loss or Loss in Short Flows

When the lost segments are at the end of a transaction, TCP's fast retransmit algorithm does not work as there are no ACKs to trigger it. When a sender does not receive an ACK for a given segment within a certain amount of time called retransmission timeout (RTO), it re-sends the segment [RFC6298]. RTO can be as long as several seconds. Hence the recovery of lost segments triggered by RTO is lengthy. [I-D.dukkipati-tcpm-tcp-loss-probe] indicates that large RTOs make a significant contribution to the long tail on the latency statistics of short flows such as loading web pages.

The short-lived flows often complete in one or two RTTs. Even when the lost packet is not an exact tail, it can possibly add another RTT

because there may not be enough packets in flight to trigger the fast retransmit). In long-haul networks, it can result in extra time of tens or hundreds of milliseconds. For ant short lived or transactional flows, it affects the latency greatly.

An overlay segment transmits the aggregated flows from ON to ON. As short-lived flows are aggregated, the probability of tail loss over this specific overlay segment decreases compared to an individual flow. The overlay segment is much shorter than the end-to-end path, hence loss recovery over an overlay segment helps to obtain low latency.

4.2. Packet Loss in Real Time Media Streams

The Real-time transport protocol (RTP) is widely used in interactive audio and video. Packet loss degrades the quality of the received media. When the latency tolerance of the application is sufficiently large, the RTP sender may use RTCP NACK feedback from the receiver [RFC4585] to trigger the retransmission of the lost packets before the playout time is reached at the receiver.

The end-to-end path over WAN can be hundreds of milliseconds, so the end-to-end feedback based retransmission may be not be very useful when applications can not tolerate one more RTT. Loss recovery over an overlay segment can then be used for the scenarios in which a shorter delayed retransmission can catch up with the playout time.

5. Features to be Considered for LOOPS

This section provides an overview of the LOOPS features. This section is not meant to document a detailed specification, but it is meant to highlight some design choices that may be followed during the solution design phase.

5.1. Local Recovery

LOOPS (Local Optimizations on Path Segments) aims to provide in-network recovery over segments to achieve better data delivery by making packet loss recovery faster. This is viable because LOOPS nodes will be instantiated to partition the path into segments. At the same time, LOOPS does not replace the end-to-end loss recovery (if any). With the advent of automation and technologies like NFV and virtual IO, it is possible to dynamically instantiate functions to nodes. The enabling of LOOPS is expected to be dynamic. When to enable this function is out of scope. The operator or administrator can make the decision based on their historical experience or real-time monitoring.

There are two ways to recover packet, retransmission and Forward Error Correction (FEC). A document to specify the generic elements for loss detection, sequence number space, acknowledgment generation and state transition is available in [I-D.welzl-loops-gen-info].

5.2. Congestion Control Interaction

When a TCP-like transport layer protocol is used, local recovery in LOOPS has to interact with the upper layer transport congestion control. Classic TCP adjusts the congestion window when a loss is detected and then fast retransmit is invoked. LOOPS performs in-network recovery which may cause a loss event not being observed by the TCP sender. Then TCP sender may overshoot then.

To solve this issue, LOOPS needs to report the loss to end-to-end congestion control LOOPS. LOOPS can CE(Congestion Experienced) marks its recovered packets as the loss signal to end-to-end. Converting a packet loss signal to CE marking signal brings the benefits of reducing Head-of-Line blocking and probability of RTO expiry [RFC8087] without affecting TCP sender's loss based congestion control behaviour while enjoying the faster local recovery. ECN based indication is equivalent to a loss event at the TCP sender [RFC3168]. In this way, a requirement is set for applying LOOPS. Only ECT (ECN-Capable Transport) flows should be directed to an LOOPS enabled path segment.

5.3. Overlay Protocol Extensions

Some tunnel protocols such as VXLAN [RFC7348], GENEVE [I-D.ietf-nvo3-geneve], LISP [RFC6830] or CAPWAP [RFC5415] are employed in overlay network. They are used in various ways. A path can have single overlay tunnel as a sub-path or stitch multiple segments together, like VXLAN [RFC7348] or GENEVE [I-D.ietf-nvo3-geneve], or specify a sequence of intermediate nodes, as in SRv6 [RFC8754].

LOOPS does not look into the inner packet. LOOPS information is required to be embedded in the overlay protocol header. An example shown in Figure 4. The current protocol focus is GENEVE [I-D.ietf-nvo3-geneve]. The specific information is to be defined in separate documents.

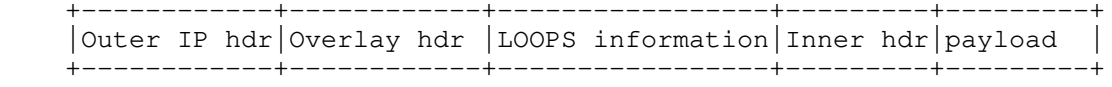


Figure 4: LOOPS Extension Header Example

6. Local in-network Recovery and End-to-end Retransmission

Most transport layer protocols have their own end-to-end retransmission to recover the lost packet. When LOOPS is in use, its local recovery can affect the end-to-end one. This section talks about such impacts.

There are two ways to perform local recovery, retransmission and FEC (Forward Error Correction). They are possibly used together in some cases. Such approaches between two overlay nodes recover the lost packet in relatively shorter distance and thus shorter latency. Therefore the local recovery is generally faster compared to end-to-end.

End-to-end retransmission is normally triggered by a NACK as in RTCP, multiple duplicate ACKs as in traditional TCP or time based detection as in RACK [I-D.ietf-tcpm-rack].

When FEC is used for local recovery, it may come with a buffer to make sure the recovered packets delivered are in order subsequently. Therefore the receiver side is unlikely to see the out-of-order packets and then send a NACK or multiple duplicate ACKs. The side effect to unnecessarily trigger end-to-end retransmit is minimum. When FEC is used in this way, if redundancy and block size are determined, extra latency required to recover lost packets is also bounded. Then RTT variation caused by it is predictable. In some extreme case like a large number of packet loss caused by persistent burst, FEC may not be able to recover it. Then end-to-end retransmit will work as a last resort. In summary, when FEC is used as local recovery, the impact on end-to-end retransmission is limited.

When local retransmission is used, it has the following impacts on the end-to-end retransmission.

For packet loss in RTP streaming, local retransmission can recover those packets which would not be retransmitted end-to-end otherwise due to long RTT. Therefore when the segment(s) being retransmitted on is a small portion of the whole end to end path, the retransmission will have a significant effect of improving the quality at receiver.

When the sender also re-transmits the packet based on a NACK received, the receiver may receive the duplicated retransmitted packets.

For packet loss in TCP flows, TCP RENO and CUBIC use duplicate ACKs as a loss signal to trigger the fast retransmit. Though we are not

standardize the buffering feature of a LOOPS egress, an introductory analysis is given as follows.

- The egress overlay node can buffer the out-of-order packets for a while, giving a limited time for a packet being retransmitted somewhere in the overlay path to reach it. The retransmitted packet and the buffered packets caused by it may increase the RTT variation at the sender. When the retransmitted latency is a small portion of RTT or the loss is rare, such RTT variation will be smoothed without much impact.

  The buffer management is nontrivial in this case. It has to be determined how many out-of-order packets can be buffered at the egress overlay node before it gives up waiting for a successful local retransmission. In some extreme case the lost packet is not recovered successfully locally, the sender may invoke end-to-end fast retransmit slower than it would be in classic TCP.

- If LOOPS network does not buffer the out-of-order packets caused by packet loss, TCP sender which uses a time based loss detection like RACK [I-D.ietf-tcpm-rack] will perform well here. It uses the notion of time to replace the conventional DUPACK threshold approach to detect losses. Hence it prevents the TCP sender from invoking fast retransmit too early. Local retransmission will not interfere the sender's retransmission generally in this case. If time based loss detection is not supported at the sender, end to end retransmission may be invoked as usual. It consumes extra bandwidth Because the lost packets (i.e. recovered packet) is normally a very small percentage of the total packets. Then extra bandwidth cost is not significant.

7. Summary

LOOPS will extend the existing overlay protocols in data plane, potential starting from GENEVE [I-D.ietf-nvo3-geneve] which has good extensibility. Path or segment selection can be feature provided by the overlay protocols via SDN techniques [RFC7149] or other approaches and is not a part of LOOPS. LOOPS is a set of functions to be implemented on Overlay Nodes as a tunnel transport with best effort reliability. LOOPS targets the following features.

1. Local recovery: Local recovery: Retransmission, FEC, or combination thereof can be used as local recovery method. Such recovery mechanism is in-network. It is performed by two network nodes with computing and memory resources.

2. Local measurement: Ingress/Egress overlay nodes measure the local segment RTT, loss and/or throughput to immediately get the overlay segment status for loss detection.

3. Interact with end-to-end congestion control: Convert a packet loss signal to an ECN-marking signal to notify the end host sender.

8. Security Considerations

LOOPS does not require access to the traffic payload in clear, so encrypted payload does not affect functionality of LOOPS.

The use of LOOPS introduces some issues which impact security. ON with LOOPS function represents a point in the network where the traffic can be potentially manipulated and intercepted by malicious nodes. Means to ensure that only legitimate nodes are involved should be considered.

Denial of service attack can be launched from an ON. A rogue ON might be able to spoof packets as if it come from a legitimate ON. It may also modify the ECN CE marking in packets to influence the sender's rate. In order to protected from such attacks, the overlay protocol itself should have some built-in security protection which is used by LOOPS. The operator should use some authentication mechanism to make sure ONs are valid and non-compromised.

9. IANA Considerations

No IANA action is required.

10. Acknowledgements

Thanks to etosat mailing list about the discussion about the SatCom and LOOPS use case.

11. Informative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC3135] Border, J., Kojo, M., Griner, J., Montenegro, G., and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations", RFC 3135, DOI 10.17487/RFC3135, June 2001, <https://www.rfc-editor.org/info/rfc3135>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <https://www.rfc-editor.org/info/rfc4585>.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, DOI 10.17487/RFC4588, July 2006, <https://www.rfc-editor.org/info/rfc4588>.

[RFC5415] Calhoun, P., Ed., Montemurro, M., Ed., and D. Stanley, Ed., "Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification", RFC 5415, DOI 10.17487/RFC5415, March 2009, <https://www.rfc-editor.org/info/rfc5415>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <https://www.rfc-editor.org/info/rfc5681>.

[RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <https://www.rfc-editor.org/info/rfc6298>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <https://www.rfc-editor.org/info/rfc6830>.

[RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <https://www.rfc-editor.org/info/rfc7149>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <https://www.rfc-editor.org/info/rfc7348>.

[RFC8087] Fairhurst, G. and M. Welzl, "The Benefits of Using Explicit Congestion Notification (ECN)", RFC 8087, DOI 10.17487/RFC8087, March 2017, <https://www.rfc-editor.org/info/rfc8087>.

[RFC8517] Dolson, D., Ed., Snellman, J., Boucadair, M., Ed., and C. Jacquenet, "An Inventory of Transport-Centric Functions Provided by Middleboxes: An Operator Perspective", RFC 8517, DOI 10.17487/RFC8517, February 2019, <https://www.rfc-editor.org/info/rfc8517>.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

[I-D.dukkipati-tcpm-tcp-loss-probe] Dukkipati, N., Cardwell, N., Cheng, Y., and M. Mathis, "Tail Loss Probe (TLP): An Algorithm for Fast Recovery of Tail Losses", draft-dukkipati-tcpm-tcp-loss-probe-01 (work in progress), February 2013.

[I-D.ietf-nvo3-geneve] Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-16 (work in progress), March 2020.

[I-D.ietf-tcpm-rack] Cheng, Y., Cardwell, N., Dukkipati, N., and P. Jha, "RACK: a time-based fast loss detection algorithm for TCP", draft-ietf-tcpm-rack-08 (work in progress), March 2020.

[I-D.welzl-loops-gen-info] Welzl, M. and C. Bormann, "LOOPS Generic Information Set", draft-welzl-loops-gen-info-03 (work in progress), March 2020.

[DOI_10.1109_ICDCS.2016.49]
Cai, C., Le, F., Sun, X., Xie, G., Jamjoom, H., and R. Campbell, "CRONets: Cloud-Routed Overlay Networks", 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), DOI 10.1109/icdcs.2016.49, June 2016.

[DOI_10.1145_3038912.3052560]
Haq, O., Raja, M., and F. Dogar, "Measuring and Improving the Reliability of Wide-Area Cloud Paths", Proceedings of the 26th International Conference on World Wide Web, DOI 10.1145/3038912.3052560, April 2017.

[DOI_10.1109_INFCOMW.2019.8845208]
Xu, Z., Ju, R., Gu, L., Wang, W., Li, J., Li, F., and L. Han, "Using Overlay Cloud Network to Accelerate Global Communications", IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), DOI 10.1109/infcomw.2019.8845208, April 2019.

Authors' Addresses

Yizhou Li
Huawei Technologies

Email: liyizhou@huawei.com

Xingwang Zhou
Huawei Technologies

Email: zhouxingwang@huawei.com

Mohamed Boucadair
Orange

Email: mohamed.boucadair@orange.com

Jianglong Wang
China Telecom

Email: wangjl1.bri@chinatelecom.cn

Fengwei Qin
China Mobile

Email: qinfengwei@chinamobile.com

TSVWG
Internet-Draft
Intended status: Standards Track
Expires: 14 January 2021

M. Welzl
University of Oslo
C. Bormann, Ed.
Universität Bremen TZI

13 July 2020

LOOPS Generic Information Set
draft-welzl-loops-gen-info-04

Abstract

LOOPS (Local Optimizations on Path Segments) aims to provide local (not end-to-end but in-network) recovery of lost packets to achieve better data delivery in the presence of losses. [I-D.li-tsvwg-loops-problem-opportunities] provides an overview over the problems and optimization opportunities that LOOPS could address.

The present document is a strawman for the set of information that would be interchanged in a LOOPS protocol, without already defining a specific data packet format.

The generic information set needs to be mapped to a specific encapsulation protocol to actually run the LOOPS optimizations. A companion document contains a sketch of a binding to the tunnel encapsulation protocol Geneve [I-D.ietf-nvo3-geneve].

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 https://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 14 January 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Today's networks exhibit a wide variety of data rates and, relative to those, processing power and memory capacities of nodes acting as routers. For instance, networks that employ tunneling to build overlay networks may position powerful virtual router nodes in the network to act as tunnel endpoints. The capabilities available in the more powerful cases provide new opportunities for optimizations.

LOOPS (Local Optimizations on Path Segments) aims to provide local (not end-to-end but in-network) recovery of lost packets to achieve better data delivery. [I-D.li-tsvwg-loops-problem-opportunities] provides an overview over the problems and optimization opportunities that LOOPS could address. One simplifying assumption (Section 3) in the present document is that LOOPS segments operate independently from each other, each as a pair of a LOOPS Ingress and a LOOPS Egress node.

The present document is a strawman for the set of information that would be interchanged in a LOOPS protocol between these nodes, without already defining a specific data packet format. The main body of the document defines a mode of the LOOPS protocol that is based on traditional tunneling, the "tunnel mode". Appendix B is an even rougher strawman of a radically different, alternative mode that we call "transparent mode", as well as a slightly more conventional "hybrid mode" (Appendix B.3). These different modes may be applicable to different usage scenarios and will be developed in parallel, with a view of ultimately standardizing one or more of them.

For tunnel mode, the generic information set needs to be mapped to a specific encapsulation protocol to actually run the LOOPS optimizations. LOOPS is not tied to any specific overlay protocol, but is meant to run embedded into a variety of tunnel protocols. LOOPS information is added as part of a tunnel protocol header at the LOOPS ingress as shown in Figure 1. A companion document [I-D.bormann-loops-geneve-binding] contains a sketch of a binding to the tunnel encapsulation protocol Geneve [I-D.ietf-nvo3-geneve].

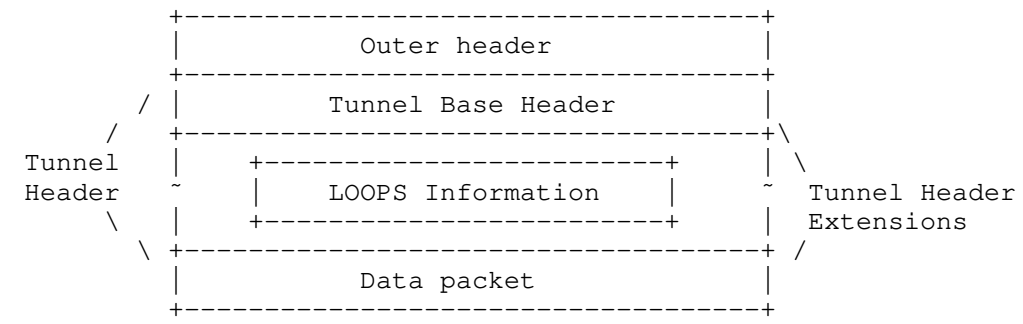


Figure 1: Packet in Tunnel with LOOPS Information

Figure 2 is extracted from the LOOPS problems and opportunities document [I-D.li-tsvwg-loops-problem-opportunities]. It illustrates the basic architecture and terms of the applicable scenario of LOOPS. Not all of the concepts introduced in the problems and opportunities document are actually used in the current strawman specification; Section 3 lays out some simplifying assumptions that the present proposal makes.

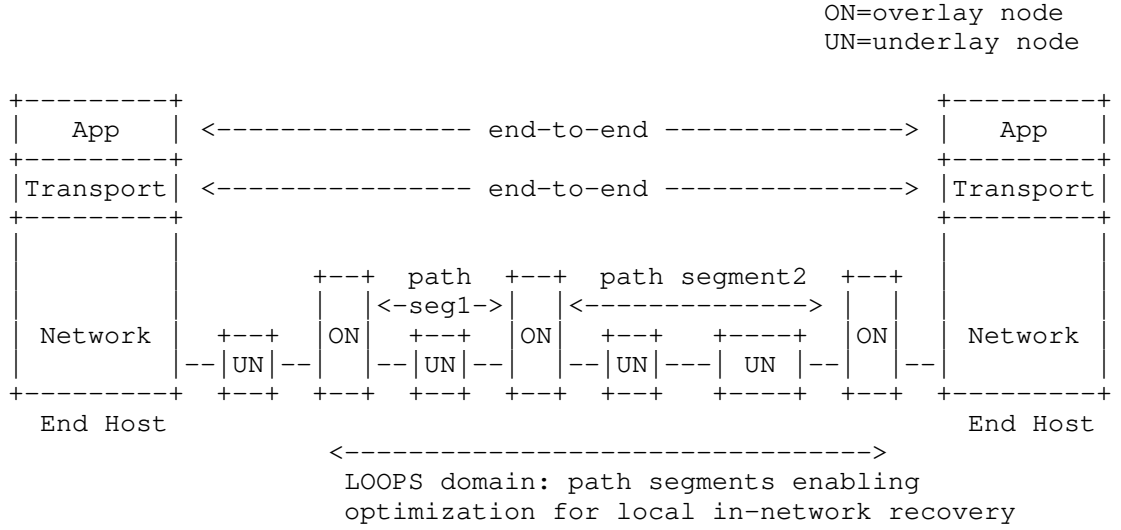


Figure 2: LOOPS Usage Scenario

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document makes use of the terminology defined in [I-D.li-tsvwg-loops-problem-opportunities]. This section defines additional terminology used by this document.

Data packets: The payload packets that enter and exit a LOOPS segment.

LOOPS Segment: A part of an end-to-end path covered by a single instance of the LOOPS protocol, the sub-path between the LOOPS Ingress and the LOOPS Egress. Several LOOPS segments may be encountered on an end-to-end path, with or without intervening routers.

LOOPS Ingress: The node that forwards data packets and forward information into the LOOPS segment, potentially performing retransmission and forward error correction based on acknowledgements and measurements received from the LOOPS Egress.

LOOPS Egress: The node that receives the data packets and forward information from the LOOPS ingress, sends acknowledgements and measurements back to the LOOPS ingress (reverse information), potentially recovers data packets from forward error correction information received.

LOOPS Nodes: Collective term for LOOPS Ingress and LOOPS Egress in a LOOPS Segment.

Forward Information: Information that is added to the stream of data packets in the forward direction by the LOOPS Ingress.

Reverse Information: Information that flows in the reverse direction, from the LOOPS Egress back to the LOOPS Ingress.

Setup Information: Information that is not transferred as part of the Forward or Reverse Information, but is part of the setup of the LOOPS Nodes.

PSN: Packet Sequence Number, a sequence number identifying a data packet between the LOOPS Ingress and Egress.

Sender: Original sender of a packet on an end-to-end path that includes one or more LOOPS segment(s).

Receiver: Ultimate receiver of a packet on an end-to-end path that includes one or more LOOPS segment(s).

2. Challenges

LOOPS has to perform well in the presence of some challenges, which are discussed in this section.

2.1. No Access to End-to-End Transport Information

LOOPS is defined to be independent of the content of the packets being forwarded: there is no dependency on transport-layer or higher information. The intention is to keep LOOPS useful with a traffic mix that may contain encrypted transport protocols such as QUIC as well as encrypted VPN traffic.

2.2. Path Asymmetry

A LOOPS segment is defined as a unidirectional forwarding path. The tunnel might be shared with a LOOPS segment in the inverse direction; this then allows to piggyback Reverse Information on encapsulated packets on that segment. But there is no guarantee that the inverse direction of any end-to-end-path crosses that segment, so the LOOPS optimizations have to be useful on their own in each direction.

2.3. Reordering vs. Spurious Retransmission

The end-to-end transport layer protocol may have its own retransmission mechanism to recover lost packets. When LOOPS recovers a loss, ideally this local recovery would replace the triggering of a retransmission at the end-to-end sender.

Whether this is possible depends on the specific end-to-end mechanism used for triggering retransmission. When end-to-end retransmission is triggered by receiving a sequence of duplicate acknowledgements (DUPACKs), and with more than a few packets in flight, the recovered packet is likely to be too late to fill the hole in the sequence number space that triggers the DUPACK detection.

(Given a reasonable setting of parameters, the local retransmission will still arrive earlier than the end-to-end retransmission and will possibly unblock application processing earlier; with spurious retransmission detection, there also will be little long-term effect on the send rate.)

While LOOPS makes no requirements on end-to-end protocols, it is worth noting that the waste of bandwidth caused by a DUPACK-based end-to-end retransmission can be avoided when the end-to-end loss detection is based on time instead of sequence numbers, e.g., with RACK [I-D.ietf-tcpm-rack]. This requires a limit on the additional latency that LOOPS will incur in its attempt to recover the loss locally. In the present version of this document, opportunity to set such a limit is provided in the Setup Information. The limit can be used to compute a deadline for retransmission, but also can be used to choose FEC parameters that keep extra latency low.

2.4. Informing the End-to-End Transport

Congestion control at the end-to-end sender is used to adapt its sending rate to the network congestion status. In typical TCP senders, packet loss implies congestion and leads to a reduction in sending rate. With LOOPS operating, packet loss can be masked from the sender as the loss may have been locally recovered. In this case, rate reduction may not be invoked at the sender. This is a desirable performance improvement if the loss was a random loss, but it is hard to ascertain that.

If LOOPS successfully conceals congestion losses from the end-to-end transport protocol, that might increase the rate to a level that congests the LOOPS segment, or that causes excessive queueing at the LOOPS ingress. What LOOPS should be able to achieve is to let the end host sender invoke the rate reduction mechanism when there is a congestion loss no matter if the lost packet was recovered locally.

As with any tunneling protocol, information about congestion events inside the tunnel needs to be exported to the end-to-end path the tunnel is part of. See e.g., [RFC6040] for a discussion of how to do this in the presence of ECN. A more recent draft, [I-D.ietf-tsvwg-tunnel-congestion-feedback], proposes to activate ECN for the tunnel regardless of whether the end-to-end protocol signals the use of an ECN-capable transport (ECT), which requires more complicated action at the tunnel egress.

A sender that interprets reordering as a signal of packet loss (DUPACKs) initiates a retransmission and reduces the sending rate. When spurious retransmission detection (e.g., via F-RTO [RFC5862] or DSACK [RFC3708]) is enabled by the TCP sender, it will often be able undo the unnecessary window reduction shortly afterwards. As LOOPS recovers lost packets locally, in most cases the end host sender will eventually find out its reordering-based retransmission (if any) is spurious. This is an appropriate performance improvement if the loss was a random loss. For congestion losses, a congestion event needs to be signaled to the end-to-end transport.

The present version of LOOPS requires the end-to-end transport to be ECN-capable (which is visible at the IP level). Congestion loss events can easily be signaled to them by setting the CE (congestion experienced) mark. Effectively, LOOPS converts a packet loss (which would be a congestion indication) to a CE mark (which also is a congestion indication).

In effect, LOOPS can be used to convert a path segment that does not yet use CE marks for congestion indication, and drops packets instead, into a segment that marks for congestion and does not drop packets except in extreme cases, incurring the benefits of Using Explicit Congestion Notification (ECN) [RFC8087]. We speak about the "drop-to-mark" function of LOOPS.

3. Simplifying assumptions

The above notwithstanding, Implementations may want to make use of indicators such as transport layer port numbers to partition a tunnel flow into separate application flows, e.g., for active queue management (AQM). Any such functionality is orthogonal to the LOOPS protocol itself and thus out of scope for the present document.

One observation that simplifies the design of LOOPS in comparison to that of a reliable transport protocol is that LOOPS does not _have_ to recover every packet loss. Therefore, probabilistic approaches, and simply giving up after some time has elapsed, can simplify the protocol significantly.

For now, we assume that LOOPS segments that may line up on an end-to-end path operate independently of each other. Since the objective of LOOPS ultimately is to assist the end-to-end protocol, it is likely that some cooperation between them would be beneficial, e.g., to obtain some measurements that cover a larger part of the end-to-end path. For instance, cooperating LOOPS segments could try to divide up permissible increases to end-to-end latency between them. This is out of scope for the present version.

Another simplifying assumption is that LOOPS nodes have reasonably precise absolute time available to them, so there is no need to burden the LOOPS protocol with time synchronization. How this is achieved is out of scope.

LOOPS nodes are created and set up (information about their peers, parameters) by some control plane mechanism that is out of scope for this specification. This means there is no need in the LOOPS protocol itself to manage setup information.

4. LOOPS Architecture

From the above, the following architecture is derived for LOOPS.

LOOPS governs the segment from an ingress node to an egress node, which is part of one or more end-to-end paths. Often, a LOOPS segment will operate on aggregate traffic from many such end-to-end paths.

The LOOPS protocol itself does not define how a LOOPS segment and the protocol entities in the ingress and egress node are set up. We expect that a _setup protocol_ on the control plane will provide some _setup information_ to the two nodes, including when to start and to tear down processing.

Each LOOPS segment governs traffic on one direction in the segment. The LOOPS ingress adds _forward information_ to that traffic; the LOOPS egress removes the forward information and sends some _reverse information_ to inform the behavior of the ingress.

Hence, in the data plane, forward information is added to each data packet. Reverse information can be sent in separate packets (e.g., Geneve control-only packets [I-D.ietf-nvo3-geneve]) and/or piggybacked on a related, reverse-direction LOOPS flow, similar to the way the forward information for that flow is carried. The setup protocol is used to provide the relationship between the LOOPS segments in the two directions that is used for piggybacking reverse information.

The above describes the "tunnel mode". A transparent mode is described in Appendix B, which does not modify the data packets and therefore needs to send any forward information (if needed, e.g., for FEC) in separate packets, usually aggregated.

The LOOPS _generic information set_ defines what information is provided as setup information, forward information, and reverse information. _Bindings_ map this information set to specific control plane and data plane protocols, including defining the specific encoding being used. Where separate packets (outside the data plane protocols being used) need to be sent, a special UDP-based protocol needs to be defined as well. The various bindings aim for some commonality, so that an implementation for multiple bindings does not need to support gratuitous variety between them.

5. LOOPS Generic Information Set

This section sketches a generic information set for the LOOPS protocol. Entries marked with (*) are items that may not be necessary and probably should be left out of an initial specification.

5.1. Setup Information

Setup Information might include:

* encapsulation protocol in use, and its vital parameters

* identity of LOOPS ingress and LOOPS egress; information relevant for running the encapsulation protocol such as port numbers

* target maximum latency increase caused by the operation of LOOPS on this segment

* maximum retransmission count (*)

5.2. Forward Information

In the forward information, we have identified:

* tunnel type (a few bits, meaning agreed between Ingress and Egress)

* packet sequence number PSN (20+ bits), counting the data packets forwarded transmitted by the LOOPS ingress (i.e., retransmissions re-use the PSN)

* an "ACK desirable" flag (one bit, usually set for a certain percentage of the data packets only)

* an optional blob, to be echoed by the egress

* anything that the FEC scheme needs.

The first four together (say, 3+24+4+1) might even fit into 32 bits, but probably need up to 48 bits total. FEC info of course often needs more space.

(Note that in this proposal there is no timestamp in the forward information; see Section 6.3.)

24 bits of PSN, minus one bit for sequence number arithmetic, gives 8 million packets (or 2.4 GB at typical packet sizes) per worst-case RTT. So if that is, say, 30 seconds, this would be enough to fill 640 Mbit/s.

5.3. Reverse Information

For the reverse information, we have identified:

* one optional block 1, possibly repeated:
  - PSN being acknowledged
  - absolute time of reception for the packet acknowledged (PSN)
  - the blob, if present, echoed back
* one optional block 2, possibly repeated:
  - an ACK bitmap (based on PSN), always starting at a multiple of 8
  - a delta indicating the end PSN of the bitmap (actually the first PSN that is beyond it), using (Acked-PSN & ~7) + 8*(delta+1) as the end of the bitmap. Acked-PSN in that formula is the previous block 1 PSN seen in this packet, or 0 if none so far.

Block 1 and Block 2 can be interspersed and repeated. They can be piggybacked on a reverse direction data packet or sent separately if none occurs within some timeout. They will usually be aggregated in some useful form. Block 1 information sets are only returned for packets that have "ACK desirable" set. Block 2 information is sent by the receiver based on some saturation scheme (e.g., at least three copies for each PSN span over time). Still, it might be possible to go down to 1 or 2 amortized bytes per forward packet spent for all this.

The latency calculation is done by the sender, who occasionally sets "ACK desirable", and notes down the absolute time of transmission for this data packet (the timekeeping can be done quite efficiently as deltas). Upon reception of a block 1 ACK, it can then subtract that from the absolute time of reception indicated. This assumes time synchronization between the nodes is at least as good as the precision of latency measurement needed, which should be no problem with IEEE 1588 PTP synchronization (but could be if using NTP-based synchronization only). A sender can freely garbage collect noted down transmission time information; doing this too early just means that the quality of the RTT sampling will reduce.

6. LOOPS General Operation

In the Tunnel Mode described in the main body of this document, LOOPS information is carried by some tunnel encapsulation.

6.1. Initial Packet Sequence Number

There is no connection establishment procedure in LOOPS. The initial PSN is assigned unilaterally by the LOOPS Ingress.

Because of the short time that is usually set in the maximum latency increase, there is little damage from a collision of PSNs with packets still in flight from previous instances of LOOPS.

6.1.1. Minimizing collisions

If desired, collisions can be minimized by assigning initial PSNs randomly, or using stable storage. Random assignment is more useful for longer PSNs, where the likelihood of overlap will be low. The specific way a LOOPS ingress uses stable storage is a local matter and thus out of scope. (Implementation note: this can be made to work similar to secure nonce generation with write attenuation: Say, every 10000 packets, the sender notes down the PSN into stable storage. After a reboot, it reloads the PSN and adds 10000 in sequence number arithmetic [RFC1982], plus maybe another 10000 so the sender does not have to wait for the store operation to succeed before sending more packets.)

6.1.2. Optional Initial PSN procedure

As a potential option (to be discussed), an initial packet sequence number could be communicated using a simple two-bit protocol, based on an I flag (Initial PSN) carried in the forward information and an R flag (Initial PSN Received) in the reverse information. This procedure essentially clears the egress of any previous state, however, the benefits of this procedure are limited.

The initial PSN is assigned unilaterally by the LOOPS ingress, selected randomly. The ingress will keep setting the I flag to one when it starts to send packets from a new beginning or whenever it believes there is a need to notify the egress about a new initial PSN. The ingress will stop setting the I flag when it receives an acknowledgement with the R flag set from the egress.

When the LOOPS egress receives a packets with the I flag set, it stops performing services that assume a sequential PSN. The egress will no longer provide acknowledgement information for the packets with PSN smaller than this new initial PSN (per sequence number arithmetic [IEN74]). The egress sends acknowledgement information back without any delay by echoing the value of the I flag in the R flag. This also means the egress unsets the R flag in subsequent acknowledgements for packets with the I flag unset.

It may happen that the first few packets are lost in an initial PSN assignment process. In this case, the loss of these packets is not detectable by the LOOPS ingress since the first received PSN will be treated as an initial PSN at the egress. This is an acceptable temporary performance degradation: LOOPS does not intend to provide perfect reliability, and LOOPS usually applies to the aggregated traffic over a tunnel so that the initial PSN assignment happens infrequently.

6.2. Acknowledgement Generation

A data packet forwarded by the LOOPS ingress always carries PSN information. The LOOPS egress uses the largest newly received PSN with the "ACK desired" bit as the ACK number in the block 1 part of the acknowledgement. This means that the LOOPS ingress gets to modulate the number of acknowledgement sent by the LOOPS egress. However, whenever an out-of-order packet arrives while there still are "holes" in the PSNs received, the LOOPS receiver should generate a block 2 acknowledgement immediately that the LOOPS sender can use as an ACK list.

Reverse information can be piggybacked in a reverse direction data packet. When the reverse direction has no user data to be sent, a pure reverse information packet needs to be generated. This may be based on a short delay during which the LOOPS egress waits for a data packet to piggyback on. (To reduce MTU considerations, the egress could wait for less-than-full data packets.)

6.3. Measurement

When sending a block 1 acknowledgement, the LOOPS egress indicates the absolute time of reception of the packet. The LOOPS ingress can subtract the absolute time of transmission that it still has available, resulting in one high quality latency sample. (In an alternative design, the forward information could include the absolute time of transmission as well, and block1 information would echo it back. This trades memory management at the ingress for increased bandwidth and MTU reduction.)

When a data packet has been transmitted, it may not be clear which specific copy is acknowledged in a block 1 acknowledgement: the acknowledgement for the initial (or, more generally, an earlier) copy may have been delayed (ACK ambiguity)). The LOOPS ingress therefore SHOULD NOT base its measurements on acknowledgements for retransmitted data packets. One way to achieve this is by not setting the "ACK desired" bit on retransmissions in the first place.

The LOOPS ingress can also use the time of reception of the block 1 acknowledgement to obtain a segment RTT sample. Note that this will include any wait time the LOOPS egress incurs while waiting for a piggybacking opportunity -- this is appropriate, as all uses of an RTT will be for keeping a retransmission timeout.

To maintain quality of information during idle times, the LOOPS ingress may send keepalive packets, which are discarded at the LOOPS egress after sending acknowledgements. The indication that a packet is a keepalive packet is dependent on the encapsulation protocol.

6.3.1. Ingress-relative timestamps

As an optional procedure, the ingress node can attach a small blob of data to a forward packet that carries an ACK desired flag; this blob is then echoed by the egress in its block 1 acknowledgement. This is typically used to attach a timestamp on a time scale defined by the ingress; we speak of an ingress-relative timestamp. Alternatively, the ingress can keep a timestamp in its local storage, associated with the PSN of the packet that carries an ACK desired flag; it can then retrieve this timestamp when the block 1 acknowledgement arrives.

In either case, the LOOPS ingress keeps track of the local segment round trip time (LRTT) based on the (saved or received) timestamp and the arrival time of the block 1 acknowledgement, by setting the ACK Desired flag (D flag) occasionally (several times per RTT) and saving/including a sending timestamp for/in the packet.

As the egress will send block 1 acknowledgement information right away when it receives a packet with the D flag set, the measurement of LRTT is more accurate for such packets. A smoothed local segment round trip time S_LRTT can be computed in a similar way as defined by [RFC0793]. A recent minimum value of LRTT is also kept as min_LRTT. S_LRTT is used as a basis for the overall timing of retransmission and state management.

Retransmitted packets MUST NOT be used for local segment round trip time (LRTT) calculation.

6.3.2. ACK generation

A block 1 acknowledgement is generated based on receiving a forward packet with a D flag.

The way block 2 acknowledgement information is sent is more subject to control by the egress. Generally, the egress will aggregate ACK bits for at least K packets before sending a block 2; this can be used to amortize the overhead to close to a couple of bits per ACK. In order to counter loss of reverse information packets, an egress will also want to send an ACK bit more than once -- a saturation value of 3 or more may be chosen based on setup information. Typically, ACK bits already sent will be prepended to ACK bits that are new in this block 2 information set. If K packets do not accumulate for a while, the egress will send one or more packets with block 2 information that covers the unsent ACK bits it has so far.

(Discussion: This works best if the egress has information both about the S_RTT and min_RTT that the ingress uses and the reverse packet loss rate.)

6.4. Loss detection and Recovery

There are two ways for LOOPS local recovery, retransmission and FEC.

6.4.1. Local Retransmission

When retransmission is used as recovery mechanism, the LOOPS ingress detects a packet loss by not receiving an ACK for the packet within the time expected based on an RTO value (which might be calculated as in [RFC6298]). Packet retransmission should then not be performed more than once within an LRTT.

When a retransmission is desired, the LOOPS ingress performs the local in-network recovery by retransmitting the packet. Further retransmissions may be desirable if the lack of ACK is persistent beyond an RTO, as long as the maximum latency increase is not reached.

6.4.2. FEC

FEC is another way to perform local recovery. When FEC is in use, a FEC header is sent with data packets as well as with special repair packets added to the flow. The specific FEC scheme used could be defined in the Setup Information, using a mechanism like [RFC5052]. The FEC rate (amount of redundancy added) and possibly the FEC scheme could be unilaterally adjusted by the LOOPS ingress in an adaptive mechanism based on the measurement information.

6.5. Discussion

Without progress in the way that end-host transport protocols handle reordering, LOOPS will be unable to prevent end-to-end retransmissions that duplicate effort that is spent in local retransmissions. It depends on parameters of the path segment whether this wasted effort is significant or not.

One remedy against this waste could be the introduction of resequencing at the LOOPS Egress node. This increases overall mean packet latency, but does not always increase actual end-to-end data stream latency if a head-of-line blocking transport such as TCP is in use. For applications with a large percentage of legacy TCP end-hosts and sufficient processing capabilities at the LOOPS Egress node, resequencing may be a viable choice. Note that resequencing could be switched off and on depending on some measurement information.

The packet numbering scheme chosen by LOOPS already provides the necessary information for the LOOPS Egress to reconstruct the sequence of data packets at the LOOPS ingress.

7. Sketches of Bindings to Tunnel Protocols

The LOOPS information defined above in a generic way can be mapped to specific tunnel encapsulation protocols. A sketch for the tunnel protocol Geneve is given below (Section 7.1). The actual encapsulation can be designed in a "native" way by putting each of the various elements into the TLV format of the encapsulation protocol, or it can be achieved by providing single TLVs for forward and reverse information and using some generic encoding of both kinds of information as shown in Appendix B.3.

7.1. Embedding LOOPS in Geneve

Geneve [I-D.ietf-nvo3-geneve] is an extensible overlay protocol which can embed LOOPS functions. Geneve uses TLVs to carry optional information between NVEs. NVE is logically the same entity as the LOOPS node.

The Geneve header has a mandatory Virtual Network Identifier (VNI) field. The specific VNI value to be used is part of the setup information for the LOOPS tunnel.

More details for a Geneve binding for LOOPS can be found in [I-D.bormann-loops-geneve-binding].

8. IANA Considerations

No IANA action is required at this stage. When a LOOPS representation is designed for a specific tunneling protocol, new codepoints will be required in the registries that pertain to that protocol.

9. Security Considerations

The security of a specific LOOPS segment will depend both on the properties of the generic information set described here and those of the encapsulation protocol employed. The security considerations of the encapsulation protocol will apply, as will the protection afforded by any security measures provided by the encapsulation protocol. Any LOOPS encapsulation specification is expected to provide information about preferred configurations of the encapsulation protocol employed, including security mechanisms, and to provide a security considerations section discussing the combination. The following discussion aims at discussing security considerations that will be common between different encapsulations.

9.1. Threat model

Attackers might attempt to perturb the operation of a LOOPS segment for a number of purposes:

* Denial of Service: Damaging the ability of LOOPS to recover packets, or damaging packet forwarding through the LOOPS segment in general.

* Attacks on Confidentiality or Integrity: Obtaining the content of data packets, modifying them, injecting new or suppressing specific data packets.

For the purposes of these security considerations, we can distinguish three classes of attackers:

1. on-path read-write: The attacker sees packets under way on the segment and can modify, inject, or suppress them.

   In this case there is really nothing LOOPS can do, except for acting as a full security protocol on its own, which would be the task of the encapsulation protocol. Without that, attackers already can manipulate the packet stream as they wish. This class of attackers is considered out of scope for these security considerations.

2. on-path read + inject: The attacker sees packets under way on the segment and can inject new packets.

   For this case, LOOPS itself similarly cannot add to the confidentiality of the data stream. However, LOOPS could protect against denial of service against its own protocol operation and, in a limited fashion, against attacks on integrity that wouldn't already have been possible by packet injection without LOOPS.

3. off-path inject: The attacker can inject new packets, but cannot see existing packets under way on the segment.

   Similar considerations apply as for class 2, except that the "blind" class 3 attacker might need to guess information it could have extracted from the packet stream in class 2.

9.2. Discussion

Class 2 attackers can see e.g. sequence numbers and can inject, but not modify traffic. Attacks might include injecting false ACKs, initial PSN flags, ... (TBD)

Class 3 ("blind") attackers might still be able to fake initial PSN bits + false ACKs, but will have a harder time otherwise as it would need to guess the PSN range in which it can wreak havoc. Even random guesses will sometimes hit, though, so the protocol needs to be robust to such injection attacks. ... (TBD)

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

10.2. Informative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <https://www.rfc-editor.org/info/rfc1982>.

[RFC3708] Blanton, E. and M. Allman, "Using TCP Duplicate Selective Acknowledgement (DSACKs) and Stream Control Transmission Protocol (SCTP) Duplicate Transmission Sequence Numbers (TSNs) to Detect Spurious Retransmissions", RFC 3708, DOI 10.17487/RFC3708, February 2004, <https://www.rfc-editor.org/info/rfc3708>.

[RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, DOI 10.17487/RFC5052, August 2007, <https://www.rfc-editor.org/info/rfc5052>.

[RFC5862] Yasukawa, S. and A. Farrel, "Path Computation Clients (PCC) - Path Computation Element (PCE) Requirements for Point-to-Multipoint MPLS-TE", RFC 5862, DOI 10.17487/RFC5862, June 2010, <https://www.rfc-editor.org/info/rfc5862>.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <https://www.rfc-editor.org/info/rfc6040>.

[RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <https://www.rfc-editor.org/info/rfc6298>.

[RFC6330] Luby, M., Shokrollahi, A., Watson, M., Stockhammer, T., and L. Minder, "RaptorQ Forward Error Correction Scheme for Object Delivery", RFC 6330, DOI 10.17487/RFC6330, August 2011, <https://www.rfc-editor.org/info/rfc6330>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[RFC8087] Fairhurst, G. and M. Welzl, "The Benefits of Using Explicit Congestion Notification (ECN)", RFC 8087, DOI 10.17487/RFC8087, March 2017, <https://www.rfc-editor.org/info/rfc8087>.

[I-D.ietf-tcpm-rack] Cheng, Y., Cardwell, N., Dukkipati, N., and P. Jha, "RACK: a time-based fast loss detection algorithm for TCP", Work in Progress, Internet-Draft, draft-ietf-tcpm-rack-08, 9 March 2020, <http://www.ietf.org/internet-drafts/draft-ietf-tcpm-rack-08.txt>.

[I-D.ietf-nvo3-geneve] Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", Work in Progress, Internet-Draft, draft-ietf-nvo3-geneve-16, 7 March 2020, <http://www.ietf.org/internet-drafts/draft-ietf-nvo3-geneve-16.txt>.

[I-D.li-tsvwg-loops-problem-opportunities] Yizhou, L., Zhou, X., Boucadair, M., Wang, J., and F. Qin, "LOOPS (Localized Optimizations on Path Segments) Problem Statement and Opportunities for Network-Assisted Performance Enhancement", Work in Progress, Internet-Draft, draft-li-tsvwg-loops-problem-opportunities-05, 6 July 2020, <http://www.ietf.org/internet-drafts/draft-li-tsvwg-loops-problem-opportunities-05.txt>.

[I-D.ietf-tsvwg-tunnel-congestion-feedback] Wei, X., Yizhou, L., Boutros, S., and L. Geng, "Tunnel Congestion Feedback", Work in Progress, Internet-Draft, draft-ietf-tsvwg-tunnel-congestion-feedback-07, 5 May 2019, <http://www.ietf.org/internet-drafts/draft-ietf-tsvwg-tunnel-congestion-feedback-07.txt>.

[I-D.bormann-loops-geneve-binding]
Bormann, C., "Embedding LOOPS in Geneve", Work in Progress, Internet-Draft, draft-bormann-loops-geneve-binding-01, 12 June 2020, <http://www.ietf.org/internet-drafts/draft-bormann-loops-geneve-binding-01.txt>.

[IEN74] Plummer, W.W., "Sequence Number Arithmetic", Internet Experiment Note 74, September 1978.

Appendix A. Protocol used in Prototype Implementation

This appendix describes, in a somewhat abstracted form, the protocol as used in a prototype implementation, as described by Yizhou Li, and Xingwang Zhou.

The prototype protocol can be run in one of two modes (defined by preconfiguration):

* Retransmission mode

* Forward Error Correction (FEC) mode

Forward information is piggybacked in data packets.

Reverse information can be carried in a pure acknowledgement packet or piggybacked when carrying packets for the inverse direction.

The forward information includes:

* Packet Sequence Number (PSN) (32 bits): This identifies a packet over a specific overlay segment from a specific LOOPS Ingress. If a packet is retransmitted by LOOPS, the retransmission uses the original PSN.

* Timestamp (32 bits): Information, in a format local to the LOOPS ingress, that provides the time when the packet was sent. In the current implementation, a 32-bit unsigned value specifying the time delta in some granularity from the epoch time to the sending time of the packet carrying this timestamp. The granularity can be from 1 ms to 1 second. The epoch time follows the current TCP practice which is 1 January 1970 00:00:00 UTC. Note that a retransmitted packet uses its own Timestamp.

* FEC Info for Block Code (56 bits): This header is used in FEC mode. It currently only provides for a block code FEC scheme. It includes the Source Block Number (SBN), Encoding Symbol ID (ESI), number of symbols in a single source block and symbol size. Appendix A.1 gives more details on FEC.

The reverse information includes:

* ACK Number (32 bits): The largest (in sequence number arithmetic [RFC1982]) PSN received so far.

* ACK List (variable): This indicates an array of PSN numbers to describe the PSN "holes" preceding the ACK number. It conceptually lists the PSNs of every packet perceived as lost by the LOOPS egress. In actual use, it is truncated.

* Echoed Timestamp (32 bits): The timestamp received with the packet being acknowledged.

A.1. Block Code FEC

The prototype currently uses a block code FEC scheme (RaptorQ [RFC6330]). The fields in the FEC Info forward information are:

* Source Block Number (SBN): 16 bits. An integer identifier for the source block that the encoding symbols within the packet relate to.

* Encoding Symbol ID (ESI): 16 bits. An integer identifier for the encoding symbols within the packet.

* K: 8 bits. Number of symbols in a single source block.

* T: 16 bits. Symbol size in bytes.

The LOOPS Ingress uses the data packet in Figure 1 to generate the encoding packet. Both source packets and repair packets carry the FEC header information; the LOOPS Egress reconstructs the data packets from both kinds of packets. The LOOPS Egress currently resequences the forwarded and reconstructed packets, so they are passed on in-order when the lost packets are recoverable within the source block.

The LOOPS Nodes need to agree on the use of FEC block mode and on the specific FEC Encoding ID to use; this is currently done by configuration.

Appendix B. Transparent mode

This appendix defines a very different way to provide the LOOPS services, "transparent mode". (We call the protocol described in the main body of the document "encapsulated mode".)

In transparent mode, the idea is that LOOPS does not meddle with the forward transmission of data packets, but runs on the side exchanging additional information.

An implementation could be based on conventional forwarding switches that just provide a copy of the ingress and egress packet stream to the LOOPS implementations. The LOOPS process would occasionally inject recovered packets back into the LOOPS egress node's forwarding switch, see Figure 3.

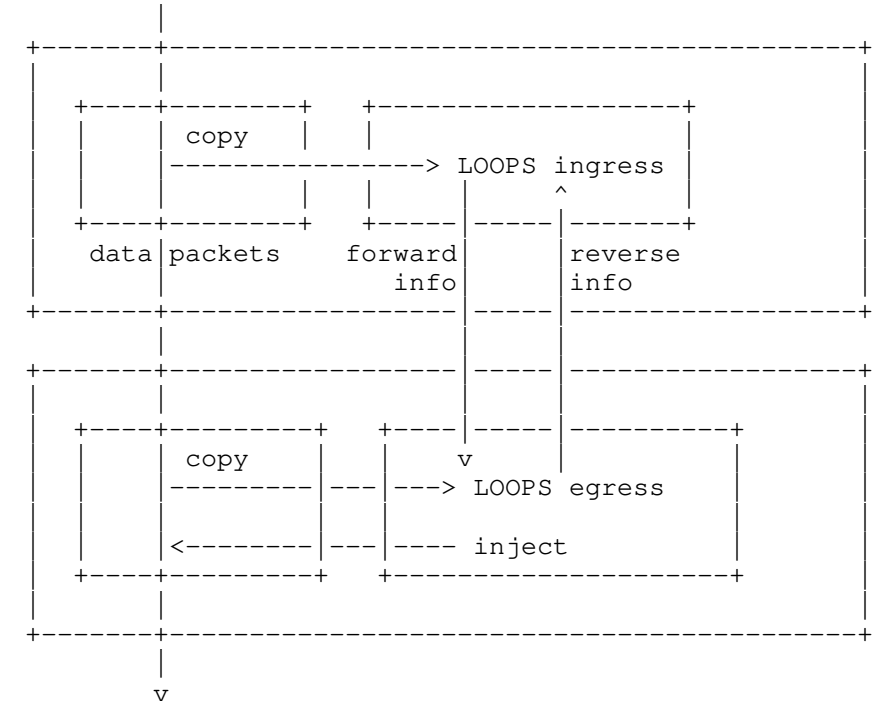


Figure 3: LOOPS Transparent Mode

The obvious advantage of transparent mode is that no encapsulation is needed, reducing processing requirements and keeping the MTU unchanged. The obvious disadvantage is that no forward information can be provided with each data packet, so a replacement needs to be found for the PSN (packet sequence number) employed in encapsulated mode. Any forward information beyond the data packets is sent in separate packets exchanged directly between the LOOPS nodes.

B.1. Packet identification

Retransmission mode and FEC mode differ in their needs for packet identification. For retransmission mode, a somewhat probabilistic accuracy of the packet identification is sufficient, for FEC mode, packet identification should not make mistakes (as these would lead to faultily reconstructed packets).

In Retransmission mode, misidentification of a packet could lead to measurement errors as well as missed retransmission opportunities. The latter will be fixed end-to-end. The tolerance for measurement errors would influence the degree of accuracy that is aimed for.

Packet identification can be based on a cryptographic hash of the packet, computed in LOOPS ingress and egress using the same algorithm (excluding fields that can change in transit, such as TTL/hop limit). The hash can directly be used as a packet number, or it can be sent in the forward information together with a packet sequence number, establishing a mapping.

For probabilistic packet identification, it is almost always sufficient to hash the first few (say, 64) bytes of the packet; all known transport protocols keep sufficient identifying information in that part (and, for encrypted protocols, the entropy will be sufficient). Any collisions of the hash could be used to disqualify the packet for measurement purposes, minimizing the measurement errors; this could allow rather short packet identifiers in retransmission mode.

For FEC mode, the packet identification together with the per-packet FEC information needs to be sent in the (separate) forward information, so that a systematic code can be reconstructed. For retransmission mode, there is no need to send any forward information for most packets, or a mapping from packet identifiers to packet sequence numbers could be sent in the forward information (probably in some aggregated form). The latter would allow keeping the acknowledgement form described in the main body (with aggregate acknowledgement); otherwise, packet identifiers need to be acknowledged. With this change, the LOOPS egress will send reverse information as in the encapsulating LOOPS protocol.

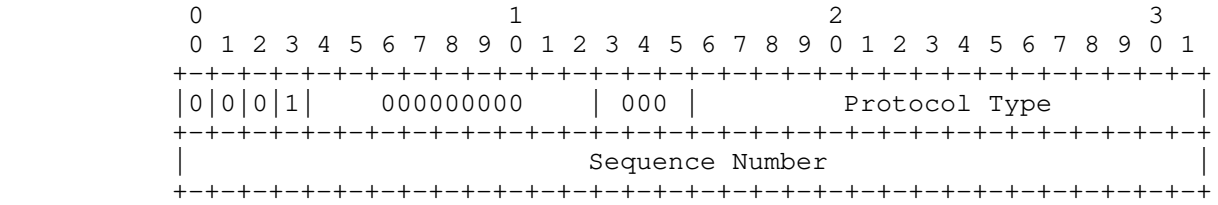
B.2. Generic information and protocol operation

With the changes outlined above, transparent mode operates just as encapsulated mode. If packet sequence numbers are not used, there is no use for block2 reverse information; if they are used, a new block3 needs to be defined that provides the mapping from packet identifiers to packet sequence numbers in the forward information. To avoid MTU reduction, some mechanism will be needed to encapsulate the actual FEC information (additional packets) in the forward information.

B.3. A hybrid mode

Figure 3 can be modified by including a GRE encapsulator into the top left corner and a GRE decapsulator in the bottom left corner. This provides more defined ingress and egress points, but it also provides an opportunity to add a packet sequence number at the ingress. The copies to the top right and bottom right corners are the encapsulated form, i.e., include the sequence number.

The GRE packet header then has the form:



The forward and reverse information can be designed closer to the approach in the main body of the document, to be exchanged using UDP packets between top right ingress and bottom right egress using a port number allocated for this purpose.

Rough ideas for both directions are given below in CDDL [RFC8610]. This information set could be encoded in CBOR or in a bespoke encoding; details such as this can be defined later.

```
forward-information = [
  [rel-psn, ack-desired, ? fec-info] /
  fec-repair-data
]

rel-psn = uint; relative packet sequence number
; always given as a delta from the previous one in the array
; starting out with a "previous value" of 0

ack-desired = bool

fec-info = [
    sbn: uint, ; Source Block Number
    esi: uint, ; Encoding Symbol ID
    ? (
      nsssb: uint; number of symbols in a single source block
      ss: uint; symbol size
    )
]

fec-repair-data = [
    repair-data: bytes
    ? (
      sbn: uint, ; Source Block Number
      esi: uint, ; Encoding Symbol ID
    )
]
```

If left out for a sequence number, the fec-info block is constructed by adding one to the previous one. fec-repair-data contain repair symbols for the sbn/esi given (which, again, are reconstructed from context if not given).

```
reverse-information = [
    block1 / block2
]

block1 = [rel-psn, timestamp]
block2 = [end-psn-delta: uint, acked-bits: bytes]
```

The acked-bits in a block2 is a bitmap that gives acknowledgments for received data packets. The bitmap always comes as a multiple of 8 bits (all bytes are filled in with 8 bits, each identifying a PSN). The end PSN of the bitmap (actually the first PSN that would be beyond it) is computed from the current PSN as set by rel-psn, rounded down to a multiple of 8, and adding 8*(end-psn-delta+1) to that value.

Acknowledgements

Sami Boutros helped with sketching the use of Geneve (Section 7.1).

Michael Welzl has been supported by the Research Council of Norway under its "Toppforsk" programme through the "OCARINA" project.

Authors' Addresses

Michael Welzl
University of Oslo
PO Box 1080 Blindern
N-0316 Oslo
Norway

Phone: +47 22 85 24 20
Email: michawe@ifi.uio.no

Carsten Bormann (editor)
Universität Bremen TZI
Postfach 330440
D-28359 Bremen
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org