

NetWork Communications Research Group (NWCRG)
Internet-Draft
Intended status: Informational
Expires: May 3, 2021

N. Kuhn, Ed.
CNES
E. Lochin, Ed.
ENAC
October 30, 2020

Network coding for satellite systems
draft-irtf-nwcrg-network-coding-satellites-15

Abstract

This document is one product of the Coding for Efficient Network Communications Research Group (NWCRG). It conforms to the directions found in the NWCRG taxonomy.

The objective is to contribute to a larger deployment of network coding techniques in and above the network layer in satellite communication systems. The document also identifies open research issues related to the deployment of network coding in satellite communication systems.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <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 May 3, 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	2
2. A Note on Satellite Networks Topology	3
3. Use-cases for Improving SATCOM System Performance Using Network Coding	5
3.1. Two-way Relay Channel Mode	5
3.2. Reliable Multicast	5
3.3. Hybrid Access	6
3.4. LAN Packet Losses	7
3.5. Varying Channel Conditions	8
3.6. Improving Gateway Handover	8
4. Research Challenges	9
4.1. Joint-use of Network Coding and Congestion Control in SATCOM Systems	9
4.2. Efficient Use of Satellite Resources	10
4.3. Interaction with Virtualized Satellite Gateways and Terminals	10
4.4. Delay/Disruption Tolerant Networking (DTN)	10
5. Conclusion	11
6. Glossary	11
7. Acknowledgements	13
8. IANA Considerations	13
9. Security Considerations	13
10. Informative References	13
Authors' Addresses	16

1. Introduction

This document is one product of and represents the collaborative work and consensus of the Coding for Efficient Network Communications Research Group (NWCRCG); while it is not an IETF product and not a standard it intends to inform the SATellite COMmunication (SATCOM) and Internet research communities about recent developments in Network Coding. A glossary is included in Section 6 to clarify the terminology use throughout the document.

As will be shown in this document, the implementation of network coding techniques above the network layer, at application or transport layers (as described in [RFC1122]), offers an opportunity for improving the end-to-end performance of SATCOM systems. While physical- and link-layer coding error protection is usually enough to

provide Quasi-Error Free transmission thus minimizing packet loss, when residual errors at those layers cause packet losses, retransmissions add significant delays (in particular in geostationary systems with over 0.7 second round-trip delays). Hence the use of network coding at the upper layers can improve the quality of service in SATCOM subnetworks and eventually favorably impact the experience of end users.

While there is an active research community working on network coding techniques above the network layer in general and in SATCOM in particular, not much of this work has been deployed in commercial systems. In this context, this document identifies opportunities for further usage of network coding in commercial SATCOM networks.

The notation used in this document is based on the NWCRG taxonomy [RFC8406]:

- o Channel and link error correcting codes are considered part of the PHYsical (PHY) layer error protection and are out of the scope of this document.
- o Forward Erasure Correction (FEC) (also called Application-Level FEC) operates above the link layer and targets packet loss recovery.
- o This document considers only coding (or coding techniques or coding schemes) that use a linear combination of packets and excludes for example content coding (e.g., to compress a video flow) or other non-linear operation.

2. A Note on Satellite Networks Topology

There are multiple SATCOM systems, for example broadcast TV, point to point communication or IoT monitoring. Therefore, depending on the purpose of the system, the associated ground segment architecture will be different. This section focuses on a satellite system that follows the European Telecommunications Standards Institute (ETSI) Digital Video Broadcasting (DVB) standards to provide broadband Internet access via ground-based gateways [ETSIEN2014]. One must note that the overall data capacity of one satellite may be higher than the capacity that one single gateway supports. Hence, there are usually multiple gateways for one unique satellite platform.

In this context, Figure 1 shows an example of a multi-gateway satellite system, where BBFRAME stands for Base-Band FRAME, PLFRAME for Physical Layer FRAME and PEP for Performance Enhancing Proxy. More information on a generic SATCOM ground segment architecture for bidirectional Internet access can be found in [SAT2017].

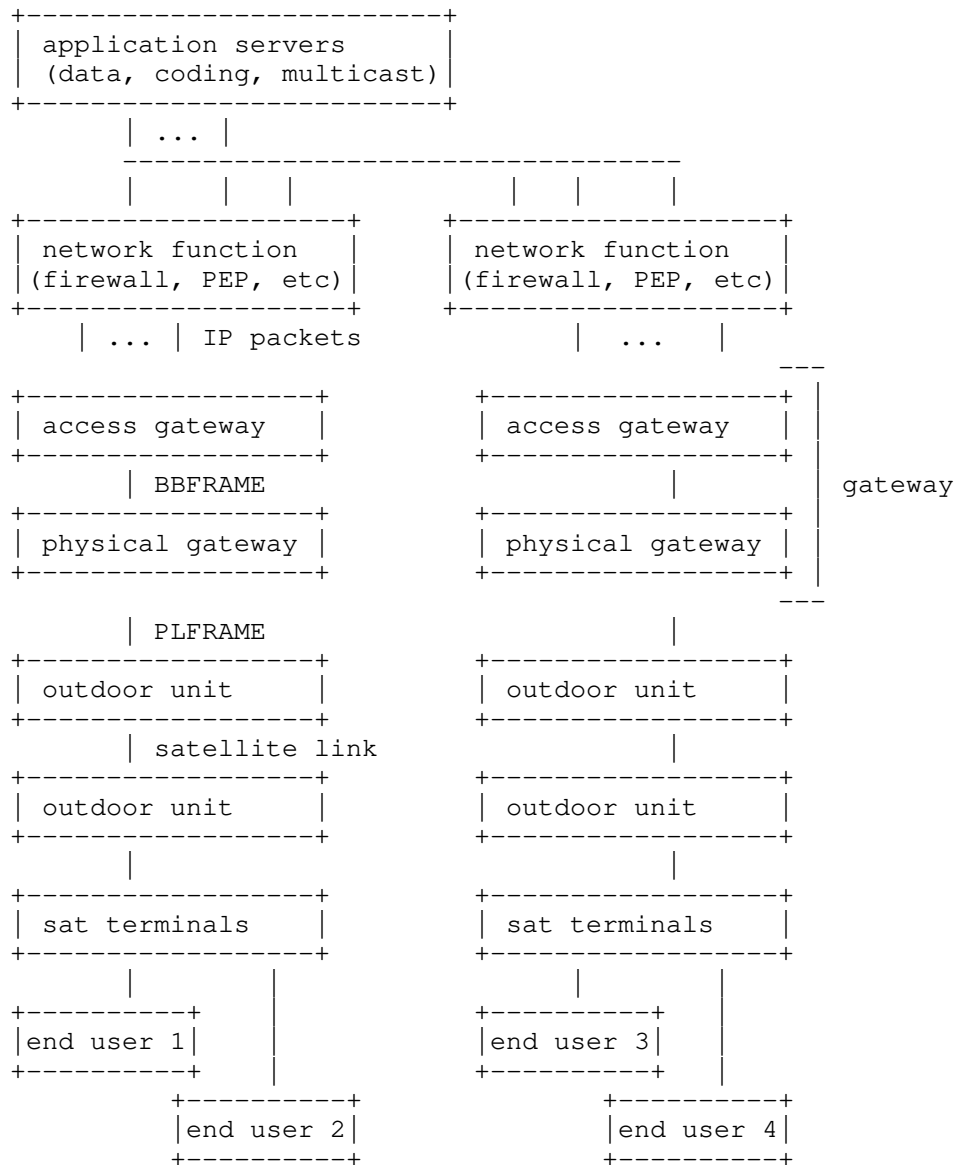


Figure 1: Data plane functions in a generic satellite multi-gateway system. More details can be found in DVB standard documents.

3. Use-cases for Improving SATCOM System Performance Using Network Coding

This section details use-cases where network coding techniques could improve SATCOM system performance.

3.1. Two-way Relay Channel Mode

This use-case considers two-way communication between end-users, through a satellite link as seen in Figure 2.

Satellite terminal A sends a packet flow A and satellite terminal B sends a packet flow B to a coding server. The coding server then sends a combination of both flows instead of each individual flows. This results in non-negligible capacity savings that has been demonstrated in the past [ASMS2010]. In the example, a dedicated coding server is introduced (note that its location could be different based on deployment use-case). The network coding operations could also be done at the satellite level, although this would require a lot of computational resources on-board and may not be supported by today's satellites.

-X}- : traffic from satellite terminal X to the server
 ={X+Y}= : traffic from X and Y combined sent from
 the server to terminals X and Y

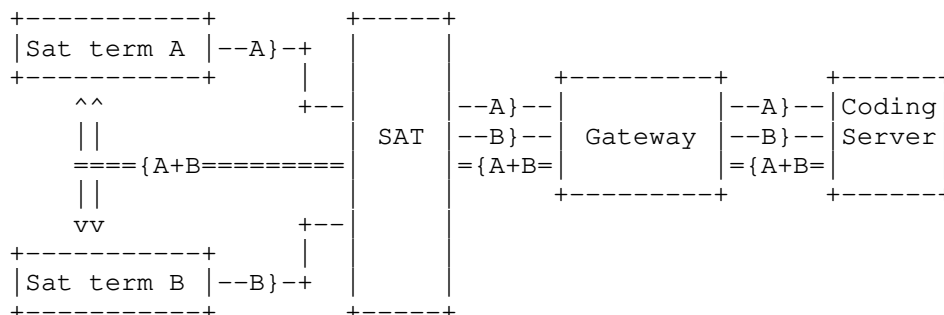


Figure 2: Network Architecture for Two-way Relay Channel using NC

3.2. Reliable Multicast

The use of multicast servers is one way to better utilize satellite broadcast capabilities. As one example satellite-based multicast is proposed in the SHINE ESA project [I-D.vazquez-nfvrg-netcod-function-virtualization] [SHINE]. This use-case considers adding redundancy to a multicast flow depending on what has been received by different end-users, resulting in non-

negligible savings of the scarce SATCOM resources. This scenario is shown in Figure 3.

-Li}- : packet indicating the loss of packet i of a multicast flow M
 =M== : multicast flow including the missing packets

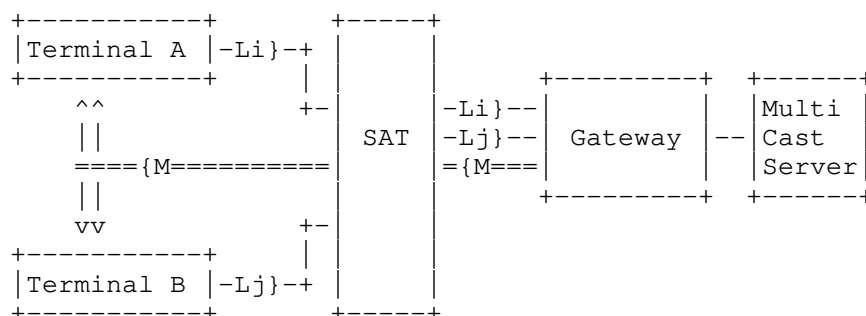


Figure 3: Network Architecture for a Reliable Multicast using NC

A multicast flow (M) is forwarded to both satellite terminals A and B. However packet Ni (respectively Nj) gets lost at terminal A (respectively B), and terminal A (respectively B) returns a negative acknowledgment Li (respectively Lj), indicating that the packet is missing. Using coding, either the access gateway or the multicast server can include a repair packet (rather than the individual Ni and Nj packets) in the multicast flow to let both terminals recover from losses.

This could also be achieved by using other multicast or broadcast systems, such as NACK-Oriented Reliable Multicast (NORM) [RFC5740] or File Delivery over Unidirectional Transport (FLUTE) [RFC6726]. Both NORM and FLUTE are limited to block coding; neither of them support more flexible sliding window encoding schemes that allow decoding before receiving the whole block an added delay benefit [RFC8406][RFC8681].

3.3. Hybrid Access

This use-case considers improving multiple path communications with network coding at the transport layer (see Figure 4, where DSL stands for Digital Subscriber Line, LTE for Long Term Evolution and SAT for SATellite). This use-case is inspired by the Broadband Access via Integrated Terrestrial Satellite Systems (BATS) project and has been published as an ETSI Technical Report [ETSITR2017].

To cope with packet loss (due to either end-user mobility or physical-layer residual errors), network coding can be introduced.

Depending on the protocol, network coding could be applied at each of the Customer Premises Equipment (CPE) and at the concentrator or both. Apart from packet losses, other gains from this approach include a better tolerance to out-of-order packet delivery which occur when exploited links exhibit high asymmetry in terms of Round-Trip Time (RTT). Depending on the ground architecture [I-D.chin-nfvrg-cloud-5g-core-structure-yang] [SAT2017], some ground equipment might be hosting both SATCOM and cellular network functionality.

-{}- : bidirectional link

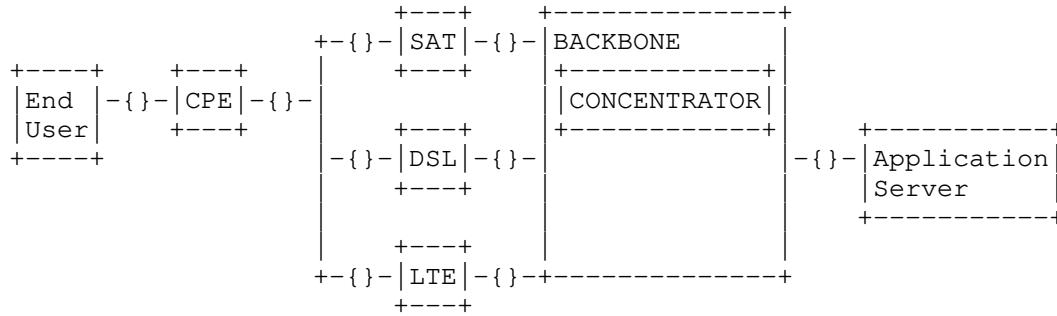


Figure 4: Network Architecture for a Hybrid Access Using Network Coding

3.4. LAN Packet Losses

This use-case considers using network coding in the scenario where a lossy WIFI link is used to connect to the SATCOM network. When encrypted end-to-end applications based on UDP are used, a Performance Enhancing Proxy (PEP) cannot operate hence other mechanism need to be used. The WIFI packet losses will result in an end-to-end retransmission that will harm the end-user quality of experience and poorly utilize SATCOM bottleneck resource for non-revenue generating traffic. In this use-case, adding network coding techniques will prevent the end-to-end retransmission from occurring since the packet losses would probably be recovered.

The architecture is shown in Figure 5.

-{}- : bidirectional link
 -''- : Wi-Fi link
 C : where network coding techniques could be introduced

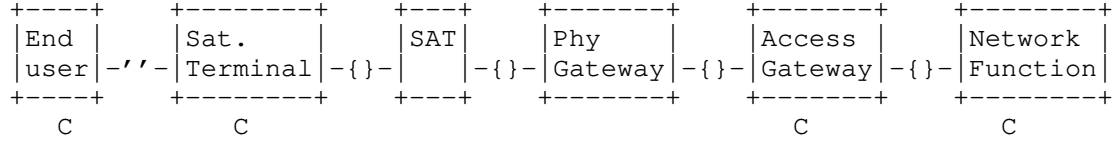


Figure 5: Network Architecture for dealing with LAN Losses

3.5. Varying Channel Conditions

This use-case considers the usage of network coding to cope with sub second physical channel condition changes where the physical-layer mechanisms (Adaptive Coding and Modulation (ACM)) may not adapt the modulation and error-correction coding in time: the residual errors lead to higher layer packet losses that can be recovered with network coding. This use-case is mostly relevant when mobile users are considered or when the satellite frequency band introduces quick changes in channel condition (Q/V bands, Ka band, etc.). Depending on the use-case (e.g., very high frequency bands, mobile users), the relevance of adding network coding is different.

The system architecture is shown in Figure 6.

-{}- : bidirectional link
 C : where network coding techniques could be introduced

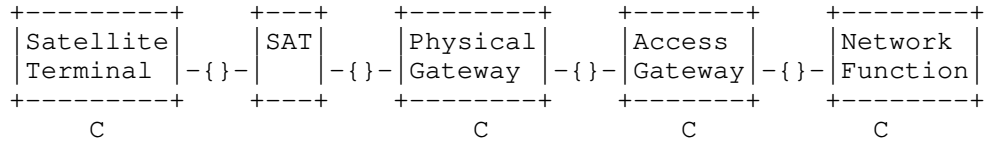


Figure 6: Network Architecture for dealing with Varying Link Characteristics

3.6. Improving Gateway Handover

This use-case considers the recovery of packets that may be lost during gateway handover. Whether for off-loading a given equipment or because the transmission quality differs from gateway to gateway, switching the transmission gateway may be beneficial. However, packet losses can occur if the gateways are not properly synchronized or if the algorithm used to trigger gateway handover is not properly tuned. During these critical phases, network coding can be added to

improve the reliability of the transmission and allow a seamless gateway handover.

Figure 7 illustrates this use-case.

-{}- : bidirectional link

! : management interface

C : where network coding techniques could be introduced

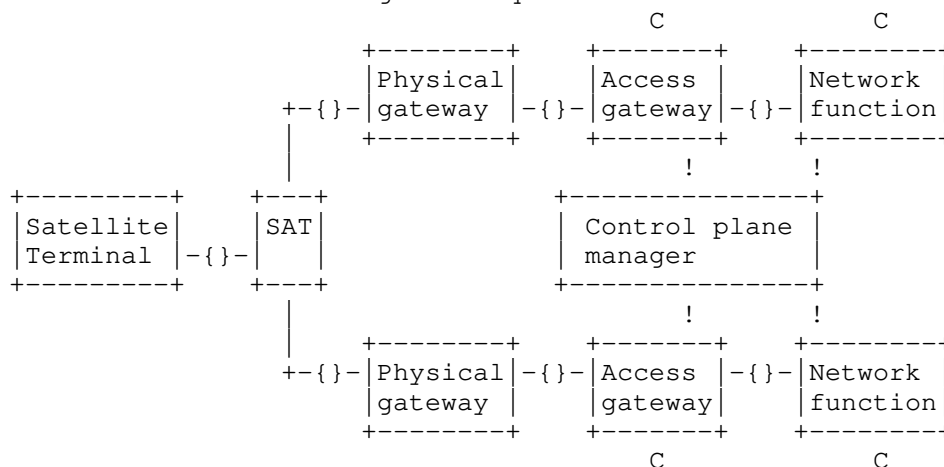


Figure 7: Network Architecture for dealing with Gateway Handover

4. Research Challenges

This section proposes a few potential approaches to introduce and use network coding in SATCOM systems.

4.1. Joint-use of Network Coding and Congestion Control in SATCOM Systems

Many SATCOM systems typically use Performance Enhancing Proxy (PEP) RFC 3135 [RFC3135]. PEPs usually split end-to-end connections and forward transport or application layer packets to the satellite baseband gateway. PEPs contribute to mitigate congestion in a SATCOM systems by limiting the impact of long delays on Internet protocols. A PEP mechanism could also include network coding operation and thus support the use-cases that have been discussed in the Section 3 of this document.

Deploying network coding in the PEP could be relevant and be independent from the specifics of a SATCOM link. This however leads to research questions dealing with the potential interaction between

network coding and congestion control. This is discussed in [I-D.irtf-nwcr-g-coding-and-congestion].

4.2. Efficient Use of Satellite Resources

There is a recurrent trade-off in SATCOM systems: how much overhead from redundant reliability packets can be introduced to guarantee a better end-user QoE while optimizing capacity usage? At which layer this supplementary redundancy should be added?

This problem has been tackled in the past by the deployment of physical-layer error-correction codes, but there remains questions on adapting the coding overhead and added delay for, e.g., the quickly varying channel conditions use-case where ACM may not be reacting quickly enough as was discussed in Section 3.5. The higher layer with network coding does not react more quickly than the physical layer, but may operate over a packet-based time window that is larger than the physical one.

4.3. Interaction with Virtualized Satellite Gateways and Terminals

In the emerging virtualized network infrastructure, network coding could be easily deployed as Virtual Network Functions (VNF). The next generation of SATCOM ground segments will rely on a virtualized environment to integrate to terrestrial networks. This trend towards Network Function Virtualization (NFV) is also central to 5G and next generation cellular networks, making this research applicable to other deployment scenarios [I-D.chin-nfvrg-cloud-5g-core-structure-yang]. As one example, the network coding VNF deployment in a virtualized environment has been presented in [I-D.vazquez-nfvrg-netcod-function-virtualization].

A research challenge would be the optimization of the NFV service function chaining, considering a virtualized infrastructure and other SATCOM specific functions, in order to guarantee efficient radio-link usage and provide easy-to-deploy SATCOM services. Moreover, another challenge related to a virtualized SATCOM equipment is the management of limited buffered capacities in large gateways.

4.4. Delay/Disruption Tolerant Networking (DTN)

Communications among deep-space platforms and terrestrial gateways can be a challenge. Reliable end-to-end (E2E) communications over such paths must cope with very long delays and frequent link disruptions; indeed, E2E connectivity may only be available intermittently, if at all. Delay/Disruption Tolerant Networking (DTN) [RFC4838] is a solution to enable reliable internetworking space communications where both standard ad-hoc routing and E2E

Internet protocols cannot be used. Moreover, DTN can also be seen as an alternative solution to transfer data between a central PEP and a remote PEP.

Network Coding enables E2E reliable communications over a DTN with potential adaptive re-encoding, as proposed in [THAI15]. Here, the use-cases proposed in Section 3.5 would encourage the usage of network coding within the DTN stack to improve the physical channel utilization and minimize the effects of the E2E transmission delays. In this context, the use of packet erasure coding techniques inside a Consultative Committee for Space Data Systems (CCSDS) architecture has been specified in [CCSDS-131.5-0-1]. One research challenge remains on how such network coding can be integrated in the IETF DTN stack.

5. Conclusion

This document introduces some wide-scale network coding technique opportunities in satellite telecommunications systems.

Even though this document focuses on satellite systems, it is worth pointing out that some scenarios proposed here may be relevant to other wireless telecommunication systems. As one example, the generic architecture proposed in Figure 1 may be mapped onto cellular networks as follows: the 'network function' block gathers some of the functions of the Evolved Packet Core subsystem, while the 'access gateway' and 'physical gateway' blocks gather the same type of functions as the Universal Mobile Terrestrial Radio Access Network. This mapping extends the opportunities identified in this document since they may also be relevant for cellular networks.

6. Glossary

The glossary of this memo extends the glossary of the taxonomy document [RFC8406] as follows:

- o ACM : Adaptive Coding and Modulation;
- o BBFRAME: Base-Band FRAME - satellite communication layer 2 encapsulation work as follows: (1) each layer 3 packet is encapsulated with a Generic Stream Encapsulation (GSE) mechanism, (2) GSE packets are gathered to create BBFRAMEs, (3) BBFRAMEs contain information related to how they have to be modulated (4) BBFRAMEs are forwarded to the physical-layer;
- o CPE: Customer Premises Equipment;
- o COM: COMMunication;

- o DSL: Digital Subscriber Line;
- o DTN: Delay/Disruption Tolerant Networking;
- o DVB: Digital Video Broadcasting;
- o E2E: End-to-end;
- o ETSI: European Telecommunications Standards Institute;
- o FEC: Forward Erasure Correction;
- o FLUTE: File Delivery over Unidirectional Transport [RFC6726];
- o IntraF: Intra-Flow Coding;
- o InterF: Inter-Flow Coding;
- o IoT: Internet of Things;
- o LTE: Long Term Evolution;
- o MPC: Multi-Path Coding;
- o NC: Network Coding;
- o NFV: Network Function Virtualization - concept of running software-defined network functions;
- o NORM: NACK-Oriented Reliable Multicast [RFC5740];
- o PEP: Performance Enhancing Proxy [RFC3135] - a typical PEP for satellite communications include compression, caching and TCP ACK spoofing and specific congestion control tuning;
- o PLFRAME: Physical Layer FRAME - modulated version of a BBFRAME with additional information (e.g., related to synchronization);
- o QEF: Quasi-Error-Free;
- o QoE: Quality-of-Experience;
- o QoS: Quality-of-Service;
- o RTT: Round-Trip Time;
- o SAT: SATellite;

- o SATCOM: generic term related to all kinds of SATellite COMMunication systems;
- o SPC: Single-Path Coding;
- o VNF: Virtual Network Function - implementation of a network function using software.

7. Acknowledgements

Many thanks to John Border, Stuart Card, Tomaso de Cola, Vincent Roca, Lloyd Wood and Marie-Jose Montpetit for their help in writing this document.

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

Security considerations are inherent to any access network, and in particular SATCOM systems. Such as it is done in cellular networks, over-the-air data can be encrypted using e.g. [ETSITS2011]. Because the operator may not enable this [SSP-2020], the applications should apply cryptographic protection. The use of FEC or Network Coding in SATCOM comes with risks (e.g., a single corrupted redundant packet may propagate to several flows when they are protected together in an Inter-Flow coding approach, see section Section 3). While this document does not further elaborate on this, the security considerations discussed in [RFC6363] apply.

10. Informative References

[ASMS2010]

De Cola, T. and et. al., "Demonstration at opening session of ASMS 2010", Advanced Satellite Multimedia Systems (ASMS) Conference , 2010.

[CCSDS-131.5-0-1]

"Erasure correcting codes for use in near-earth and deep-space communications", CCSDS Experimental specification 131.5-0-1, 2014.

[ETSIEN2014]

"Digital Video Broadcasting (DVB); Second Generation DVB Interactive Satellite System (DVB-RCS2); Part 2: Lower Layers for Satellite standard", ETSI EN 301 545-2, 2014.

- [ETSI TR2017]
"Satellite Earth Stations and Systems (SES); Multi-link routing scheme in hybrid access network with heterogeneous links", ETSI TR 103 351, 2017.
- [ETSI TS2011]
"Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 5: CPCM Security Toolbox", ETSI TS 102 825-5, 2011.
- [I-D.chin-nfvrg-cloud-5g-core-structure-yang]
Chen, C. and Z. Pan, "Yang Data Model for Cloud Native 5G Core structure", draft-chin-nfvrg-cloud-5g-core-structure-yang-00 (work in progress), December 2017.
- [I-D.irtf-nwcr-g-coding-and-congestion]
Kuhn, N., Lochin, E., Michel, F., and M. Welzl, "Coding and congestion control in transport", draft-irtf-nwcr-g-coding-and-congestion-03 (work in progress), July 2020.
- [I-D.vazquez-nfvrg-netcod-function-virtualization]
Vazquez-Castro, M., Do-Duy, T., Romano, S., and A. Tulino, "Network Coding Function Virtualization", draft-vazquez-nfvrg-netcod-function-virtualization-02 (work in progress), November 2017.
- [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <<https://www.rfc-editor.org/info/rfc1122>>.
- [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>>.
- [RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, DOI 10.17487/RFC4838, April 2007, <<https://www.rfc-editor.org/info/rfc4838>>.
- [RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", RFC 5740, DOI 10.17487/RFC5740, November 2009, <<https://www.rfc-editor.org/info/rfc5740>>.

- [RFC6363] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", RFC 6363, DOI 10.17487/RFC6363, October 2011, <<https://www.rfc-editor.org/info/rfc6363>>.
- [RFC6726] Paila, T., Walsh, R., Luby, M., Roca, V., and R. Lehtonen, "FLUTE - File Delivery over Unidirectional Transport", RFC 6726, DOI 10.17487/RFC6726, November 2012, <<https://www.rfc-editor.org/info/rfc6726>>.
- [RFC8406] Adamson, B., Adjih, C., Bilbao, J., Firoiu, V., Fitzek, F., Ghanem, S., Lochin, E., Masucci, A., Montpetit, M-J., Pedersen, M., Peralta, G., Roca, V., Ed., Saxena, P., and S. Sivakumar, "Taxonomy of Coding Techniques for Efficient Network Communications", RFC 8406, DOI 10.17487/RFC8406, June 2018, <<https://www.rfc-editor.org/info/rfc8406>>.
- [RFC8681] Roca, V. and B. Teibi, "Sliding Window Random Linear Code (RLC) Forward Erasure Correction (FEC) Schemes for FECFRAME", RFC 8681, DOI 10.17487/RFC8681, January 2020, <<https://www.rfc-editor.org/info/rfc8681>>.
- [SAT2017] Ahmed, T., Dubois, E., Dupe, JB., Ferrus, R., Gelard, P., and N. Kuhn, "Software-defined satellite cloud RAN", International Journal on Satellite Communications and Networking vol. 36 - <https://doi.org/10.1002/sat.1206>, 2017.
- [SHINE] Pietro Romano, S. and et. al., "Secure Hybrid In Network caching Environment (SHINE) ESA project", ESA project , 2017 on-going.
- [SSP-2020] Pavur (et al.), J., "A Tale of Sea and SkyOn the Security of Maritime VSAT Communications", IEEE Symposium on Security and Privacy 10.1109/SP40000.2020.00056, 2020.
- [THAI15] Thai, T., Chaganti, V., Lochin, E., Lacan, J., Dubois, E., and P. Gelard, "Enabling E2E reliable communications with adaptive re-encoding over delay tolerant networks", Proceedings of the IEEE International Conference on Communications <http://dx.doi.org/10.1109/ICC.2015.7248441>, June 2015.

Authors' Addresses

Nicolas Kuhn (editor)
CNES
18 avenue Edouard Belin
Toulouse 31400
France

Email: nicolas.kuhn@cnes.fr

Emmanuel Lochin (editor)
ENAC
7 avenue Edouard Belin
Toulouse 31400
France

Email: emmanuel.lochin@enac.fr

NWCRG
Internet-Draft
Intended status: Informational
Expires: April 29, 2020

N. Kuhn, Ed.
CNES
E. Lochin, Ed.
ISAE-SUPAERO
F. Michel, Ed.
UCLouvain
October 27, 2019

Coding and congestion control in transport
draft-kuhn-coding-congestion-transport-00

Abstract

There are discussions on how loss-based congestion controls consider lost packets that have been recovered by a coding mechanism. This document analyses to what extent transport protocols could ignore such signals and proposes best current practices on the interaction between congestion control and coding mechanism at the transport layer. Coding for tunnels is out-of-the scope of the document. Examples of interest for the proposed solution is to better deal with tail losses or with networks with non-congestion losses.

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 April 29, 2020.

Copyright Notice

Copyright (c) 2019 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	2
2. Base solution description	2
3. Sender-side coding solutions	3
3.1. Coded packets without considering CWND progression	3
3.2. Coded packets driven by CWND progression	3
4. Sender-side reaction to recovered packet signals	4
4.1. The sender congestion control considers recovered packet signals as congestion-implied packet losses	4
4.2. The sender adapts its window reduction to recovered packet signals	4
4.3. The sender ignores recovered packet signals	5
5. Summary	5
6. Acknowledgements	6
7. Contributors	6
8. IANA Considerations	6
9. Security Considerations	6
10. Informative References	6
Authors' Addresses	7

1. Introduction

[RFC5681] defines TCP as a loss-based congestion control. Coding mechanisms can be deployed and could hide congestion signals to the sender.

Because loss-based and delay-based congestion controls are deployed in the current Internet, this memo discusses simple best practices on how coding and congestion control mechanisms could coexist.

The proposed recommendations apply for coding at the transport layer (coding for tunnels is out-of-the scope of the document). Examples of interest for the proposed solution is to better deal with tail losses or with networks with non-congestion losses.

2. Base solution description

The base solution can be described as follows:

- o The receiver MUST indicate to the sender that one or multiple packets have been recovered using a coding scheme. Such "repaired packet signal" could be based on existing signals (even if the existing signal was not designed for that purpose, such as ECN) or on new type of signals (such as a RECOVERED frame in QUIC).
- o The sender MUST be able to detect the "recovered packet signal". The base solution does not describe how the sender reacts to such signal.

The proposed solution applies for coding in the transport layer. The proposed approach is inline with the one in [I-D.swett-nwcrg-coding-for-quic].

The proposed solution does not applies for the interaction between coding under the transport layer (i.e. not end-to-end), such as coding for tunnels.

3. Sender-side coding solutions

This section presents solutions for a sender to add application coding.

3.1. Coded packets without considering CWND progression

In this solution, the coded packets are sent on top of what is allowed by a congestion window. Examples of the solution could be adding a given percentage of the congestion window as supplementary packets or sending a given amount of coded packets at a given rate. The redundancy flow can be decorrelated from the congestion control that manages source packets : a secondary congestion control can be introduced, such as in coupled congestion control for RTP media [I-D.ietf-rmcat-coupled-cc]. An example would be to exploit a lower than best-effort congestion control [RFC6297].

The advantage of such solution is that coding would help in challenges cases where transmission losses are persistent.

The drawback of such solution is that it may result in coding solutions being unfair towards non-coding solutions. This solutions may result in adding congestion in congested networks.

3.2. Coded packets driven by CWND progression

In this solution, the coded packets are sent within what the congestion window allows, such as in [CTCP]. Examples of the solution would be sending coded packets when there is no more data to

transmit or preferably send coded packets instead of the following packets in the send buffer.

The advantage of this solution is that it does not contribute in adding more congestion than the congestion window allows. Indeed, all traffic (source and redundancy) is controlled by one congestion control only and TCP metrics for fairness can be indifferently applied in this case.

The main drawback is the decrease of goodput if coded packets are sent but are not used at the client side.

4. Sender-side reaction to recovered packet signals

Delay-based congestion controls ignore packets that have been repaired with coding. There is no need to define best current practices in this case. However, more discussions are required for congestion controls that use loss as congestion signals (potentially among other congestion detection mechanism).

4.1. The sender congestion control considers recovered packet signals as congestion-implied packet losses

In this solution, the sender reacts to recovered packet signals as to congestion-implied packet losses. That being said, this does not necessarily means that the packets have actually been lost. The server may have other means to identify that the packet was just out-of-ordered and ignore the recovered packet signals.

The advantages of the solution are (1) that coding mechanisms do not hide congestion signals, such as packets voluntarily dropped by a AQM [RFC7567] and (2) packets may be recovered faster than with traditionnal retransmission mechanisms.

The drawback of this solution is that, if there is a high non-congestion loss rate, the congestion control throughput may decrease drastically. Reporting this amount of loss to a sender may reduce the application goodput when there is no actual congestion.

4.2. The sender adapts its window reduction to recovered packet signals

In this solution, the sender does not reduce the congestion window with the same amount when the "recovered packet signal" is received, i.e. when a packet has been lost but recovered. Example of this solution could be based on [RFC8511] or considering that recovering an isolated packet is not an actual sign of congestion.

The advantage of the solution is that in cases where there is no actual congestion, coding could help in improving the transmission without ignoring congestion signals.

The main drawback is the precised design of the solution and its interaction with AQM mechanisms [RFC7567]. Moreover there may be fairness issues since AIMD convergence may not be guaranteed.

4.3. The sender ignores recovered packet signals

This is the case for delay-based congestion controls. The interaction between delay-based congestion controls and the delay induced by a coding mechanisms is an open research activity. That being said, a potential approach would be that loss-based congestion control ignores the "recovered packet signal".

The advantage of this solution is that coding would provided substantial benefits in cases where there are transmission losses.

However, this solution hides potential congestion losses, making it unfair to other congestion controls.

5. Summary

This section provides a summary on the content in previous sections. The Figure 1 sums up some recommendations. It is worth pointing out that the "coding without congestion" considers that coded packets are sent along with original data packets, in opposition with the solution where coded packets are transmitted only when there is no more original packets to transmit. Moreover, the values indicated in this Figure consider a channel that does not exhibit a high loss pattern.

Sender-side reaction to recovered packet signals	Sender-side coding solutions	
	Coding adding congestion	Coding without congestion
React as loss	fairness: ~ real-time: + bulk: ~	fairness: ++ real-time: + bulk: -
Adapt window reduction	fairness: ~ real-time: + bulk: +	fairness: + real-time: + bulk: -
Ignore signals	fairness: - real-time: + bulk: +	fairness: - real-time: + bulk: -

Figure 1: Recommendations

6. Acknowledgements

Many thanks to TBD.

7. Contributors

TBD

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

Security section goes here.

10. Informative References

[CTCP] Kim (et al.), M., "Network Coded TCP (CTCP)",
arXiv 1212.2291v3, 2013.

[I-D.ietf-rmcat-coupled-cc]
Islam, S., Welzl, M., and S. Gjessing, "Coupled congestion
control for RTP media", draft-ietf-rmcat-coupled-cc-09
(work in progress), August 2019.

- [I-D.swett-nwcrg-coding-for-quic]
Swett, I., Montpetit, M., Roca, V., and F. Michel, "Coding for QUIC", draft-swett-nwcrg-coding-for-quic-03 (work in progress), July 2019.
- [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>>.
- [RFC6297] Welzl, M. and D. Ros, "A Survey of Lower-than-Best-Effort Transport Protocols", RFC 6297, DOI 10.17487/RFC6297, June 2011, <<https://www.rfc-editor.org/info/rfc6297>>.
- [RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <<https://www.rfc-editor.org/info/rfc7567>>.
- [RFC8511] Khademi, N., Welzl, M., Armitage, G., and G. Fairhurst, "TCP Alternative Backoff with ECN (ABE)", RFC 8511, DOI 10.17487/RFC8511, December 2018, <<https://www.rfc-editor.org/info/rfc8511>>.

Authors' Addresses

Nicolas Kuhn (editor)
CNES

Email: nicolas.kuhn@cnes.fr

Emmanuel Lochin (editor)
ISAE-SUPAERO

Email: emmanuel.lochin@isae-supero.fr

Francois Michel (editor)
UCLouvain

Email: francois.michel@uclouvain.be

nwcrg
Internet-Draft
Intended status: Informational
Expires: September 10, 2020

V. Roca, Ed.
INRIA
F. Michel
UCLouvain
I. Swett
Google
M-J. Montpetit
Triangle Video
March 9, 2020

Sliding Window Random Linear Code (RLC) Forward Erasure Correction (FEC)
Schemes for QUIC
draft-roca-nwcrg-rlc-fec-scheme-for-quic-03

Abstract

This document specifies Sliding Window Random Linear Code (RLC) Forward Erasure Correction (FEC) Schemes for the QUIC transport protocol, in order to recover from packet losses.

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 September 10, 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	2
2. Definitions and Abbreviations	3
3. Procedures	3
3.1. Source Symbols Mapping	3
3.2. Source Symbols identification	4
3.3. Pseudo-Random Number Generator (PRNG)	4
3.4. Coding Coefficients Generation Function	4
4. Sliding Window RLC FEC Scheme over $GF(2^{8})$	5
4.1. Formats and Codes	5
4.1.1. Configuration Information	5
4.1.2. SRC FPI Frame Format	5
4.1.3. REPAIR Frame Format	6
4.1.4. Additional Procedures	7
4.2. FEC Code Specification	7
4.2.1. Encoding Side	7
4.2.2. Decoding Side	7
5. Security Considerations	7
6. IANA Considerations	8
7. Acknowledgments	8
8. References	8
8.1. Normative References	8
8.2. Informative References	9
Authors' Addresses	9

1. Introduction

QUIC [QUIC-transport] is a transport protocol that aims at improving network performance by enabling stream multiplexing, partial reliability, and methods of recovery besides retransmission, while also improving security. This document specifies FEC schemes for Sliding Window Random Linear Code (RLC) [RFC8681] to recover from lost packets within a single QUIC stream or across several QUIC streams, compliant with the FEC coding framework for QUIC [Coding4QUIC].

The ability to add FEC coding in QUIC may be beneficial in several situations:

- o for a robust transmission of latency-sensitive traffic, for instance real-time flows, since it enables to recover packet losses independently of the round trip time;

- o for the transmission of contents to a large set of QUIC reception endpoints, since the same repair frame may help recovering several different packet losses at different receivers;
- o for multipath communications, since repair traffic adds diversity.

2. Definitions and Abbreviations

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Terms and definitions that apply to coding are available in [nc-taxonomy]. More specifically, this document uses the following definitions:

Packet versus Symbol: a Packet is the unit of data that is exchanged over the network while a Symbol is the unit of data that is manipulated during the encoding and decoding operations

Source Symbol: a unit of data originating from the source that is used as input to encoding operations

Repair Symbol: a unit of data that is the result of a coding operation

This document uses the following abbreviations:

E: size of an encoding symbol (i.e., source or repair symbol), assumed fixed (in bytes)

3. Procedures

This section introduces the procedures that are used by these FEC Schemes.

3.1. Source Symbols Mapping

The present FEC Scheme follows the source symbols mapping specified in [Coding4QUIC]. Figure 1 illustrates this mapping.

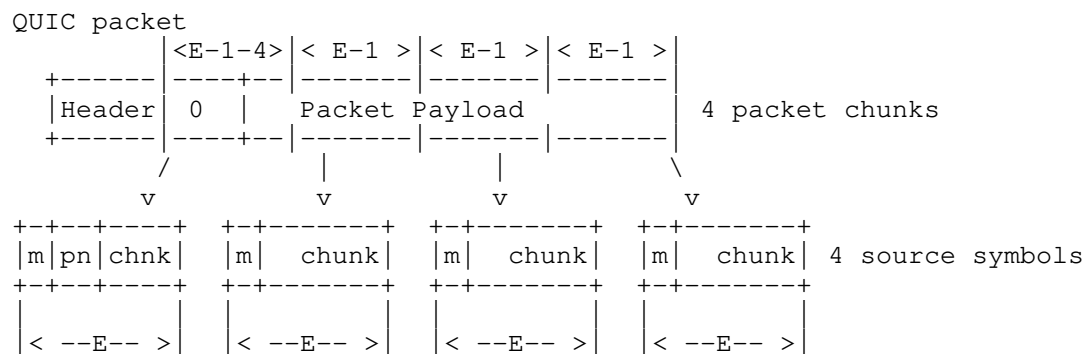


Figure 1: Example of source symbol mapping, when the E value is relatively small.

3.2. Source Symbols identification

Similarly to [RFC8681], the present FEC Scheme assigns a unique identifier (ID) to each produced source symbol. The IDs are assigned to the produced source symbols in the ascending order. The IDs start at MUST start 0 and MUST be contiguous. For any symbol with ID x , the source symbol with ID $x+1$ is :

- o The source symbol containing the next packet chunk in the same QUIC packet as the source symbol X, if it exists.
- o The source symbol containing the first packet chunk of the next generated FEC-protected QUIC packet

Do we want to authorize a wrapping of the source symbol ID ? It would be a lot easier if wrapping is not permitted.

3.3. Pseudo-Random Number Generator (PRNG)

The RLC FEC Schemes defined in this document rely on the TinyMT32 PRNG defined in [RFC8682] along with the two mapping functions to 4-bit and 8-bit unsigned integers defined in [RFC8681].

3.4. Coding Coefficients Generation Function

The coding coefficients, used during the encoding process, are generated at the RLC encoder by the `generate_coding_coefficients()` function each time a new repair symbol needs to be produced. This specification uses the `generate_coding_coefficients()` defined in [RFC8681].

4. Sliding Window RLC FEC Scheme over $GF(2^{8})$

This fully-specified FEC Scheme defines the Sliding Window Random Linear Codes (RLC) over $GF(2^{8})$.

4.1. Formats and Codes

4.1.1. Configuration Information

This section provides the RLC configuration information that needs to be shared during QUIC negotiation between the QUIC sender and receiver endpoints in order to synchronize them.

- o FEC Encoding ID (8 bits): the value assigned to this fully specified FEC Scheme MUST be XXXX, as assigned by IANA (Section 6). This FEC Encoding ID is used during the QUIC negotiation to uniquely identify the RLC FEC Scheme for QUIC;
- o Encoding symbol size, E (in bytes) (16 bits): a non-negative integer that indicates the size of each source and repair symbol, in bytes. This element is required both by the QUIC sender endpoint (RLC encoder) and the QUIC receiver endpoint(s) (RLC decoder).

TODO: specify exact format, with binary encoding, to be carried within the opaque 32-bit field during negotiation.

4.1.2. SRC FPI Frame Format

The RLC FEC Scheme requires explicit signaling of the Source Symbols it transmits. The QUIC packets whose payload is protected by FEC MUST contain an SRC FPI frame with the following format.

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               ID of First Source Symbol in Packet (i)                               ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 2: SRC FPI frame format.

The SRC FPI frame contains the ID of the first source symbol contained in this packet. Each source symbol contains a packet chunk of E-1 bytes long. If the payload to protect is longer than E-1 bytes, this means that the packet contains several packet chunks. In this case the source symbol ID will increase by exactly one for each additional packet chunk contained in the payload to protect.

Note: This frame is not idempotent. In the current version of QUIC, all the frames are idempotent (but this is not especially required). It would be great to preserve this property (a quick fix would be to add the packet number in the frame, but it takes a lot of space and I don't think it is very useful). Another bad property is that the frames are not independant anymore (several SRC FPI frames contained in the same packet have a confusing meaning). I really think that the correct solution would be a (encrypted) header field but I guess it is more complicated to propose (maybe in v2 we'll have a mechanism to define dynamic encrypted header fields during negotiation).

4.1.3. REPAIR Frame Format

The RLC FEC Scheme requires QUIC REPAIR frames to convey enough information. This section specifies the REPAIR frame format specific to the RLC FEC Scheme. Note that the notion of REPAIR frame format is equivalent to the notion of Repair FEC Payload ID in [RFC8681].

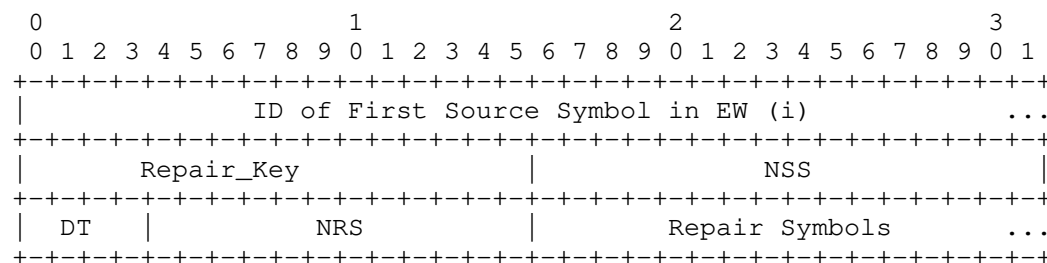


Figure 3: REPAIR frame format when protecting a single QUIC stream.

More precisely, the REPAIR frame format is composed of the following fields (Figure 3):

ID of the first Source Symbol in the Encoding Window (variable-size field):

a variable-length integer specifying the ID of the first source symbol in the encoding window. When a FEC REPAIR frame contains several repair symbols, this value applies to all of them;

Repair_Key (16-bit field): this unsigned integer is used as a seed by the coefficient generation function (Section 3.4) in order to generate the desired number of coding coefficients. When a FEC Repair Packet contains several repair symbols, this repair key value is that of the first repair symbol. The remaining repair keys can be deduced by incrementing by 1 this value, up to a maximum value of 65535 after which it loops back to 0.

Number of Source Symbols in the encoding window, NSS (16-bit field):

this unsigned integer indicates the number of source symbols in the encoding window when this repair symbol was generated. When a REPAIR frame contains several repair symbols, the NSS value applies to all of them;

Density Threshold for the coding coefficients, DT (4-bit field): this unsigned integer carries the Density Threshold (DT) used by the coding coefficient generation function Section 3.4. More precisely, it controls the probability of having a non zero coding coefficient, which equals $(DT+1) / 16$. When a REPAIR frame contains several repair symbols, the DT value applies to all of them;

Number of Repair Symbols contained in the frame, NRS (12-bit field):

this unsigned integer specifies the number of Repair Symbols contained in this REPAIR frame;

Repair Symbols: data for this repair symbol(s). This field is $NRS * E$ bytes long.

4.1.4. Additional Procedures

4.2. FEC Code Specification

This RLC FEC Scheme relies on the FEC code specification defined in [RFC8681].

4.2.1. Encoding Side

[RFC8681] high level description of a Sliding Window RLC encoder also applies here to this FEC Scheme.

4.2.2. Decoding Side

[RFC8681] high level description of a Sliding Window RLC decoder also applies here to this FEC Scheme.

5. Security Considerations

TBD

6. IANA Considerations

This document registers two values in the "QUIC FEC Encoding IDs" registry as follows:

- o XXXX refers to the Sliding Window Random Linear Codes (RLC) over $GF(2^{8})$ FEC Scheme for a Single QUIC Stream, as defined in Section 4 of this document.

7. Acknowledgments

TBD

8. References

8.1. Normative References

[Coding4QUIC]

Swett, I., Montpetit, M-J., Roca, V., and F. Michel, "Coding for QUIC", Work in Progress, NWCRG draft-swett-nwcr-g-coding-for-quic (Work in Progress), March 2020, <<https://tools.ietf.org/html/draft-swett-nwcr-g-coding-for-quic>>.

[QUIC-transport]

Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport (Work in Progress) (work in progress), November 2019, <<https://datatracker.ietf.org/doc/draft-ietf-quic-transport/>>.

[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>>.

[RFC8681] Roca, V. and B. Teibi, "Sliding Window Random Linear Code (RLC) Forward Erasure Correction (FEC) Schemes for FECFRAME", RFC 8681, DOI 10.17487/RFC8681, January 2020, <<https://www.rfc-editor.org/info/rfc8681>>.

[RFC8682] Saito, M., Matsumoto, M., Roca, V., Ed., and E. Baccelli, "TinyMT32 Pseudorandom Number Generator (PRNG)", RFC 8682, DOI 10.17487/RFC8682, January 2020, <<https://www.rfc-editor.org/info/rfc8682>>.

8.2. Informative References

[nc-taxonomy]

Roca (Ed.) et al., V., "Taxonomy of Coding Techniques for Efficient Network Communications", Request For Comments RFC 8406, June 2018, <<https://datatracker.ietf.org/doc/draft-irtf-nwcrg-network-coding-taxonomy/>>.

Authors' Addresses

Vincent Roca (editor)
INRIA
Univ. Grenoble Alpes
France

Email: vincent.roca@inria.fr

Francois Michel
UCLouvain
Louvain-la-Neuve
Belgium

Email: francois.michel@uclouvain.be

Ian Swett
Google
Cambridge, MA
US

Email: ianswett@google.com

Marie-Jose Montpetit
Triangle Video
Boston, MA
US

Email: marie@mjmontpetit.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	3
1.1. Terminology	5
2. Challenges	6
2.1. No Access to End-to-End Transport Information	6
2.2. Path Asymmetry	6
2.3. Reordering vs. Spurious Retransmission	6
2.4. Informing the End-to-End Transport	7
3. Simplifying assumptions	8
4. LOOPS Architecture	9
5. LOOPS Generic Information Set	10
5.1. Setup Information	10
5.2. Forward Information	10
5.3. Reverse Information	11
6. LOOPS General Operation	12
6.1. Initial Packet Sequence Number	12
6.1.1. Minimizing collisions	12
6.1.2. Optional Initial PSN procedure	12
6.2. Acknowledgement Generation	13
6.3. Measurement	14
6.3.1. Ingress-relative timestamps	14
6.3.2. ACK generation	15
6.4. Loss detection and Recovery	15
6.4.1. Local Retransmission	15
6.4.2. FEC	16
6.5. Discussion	16
7. Sketches of Bindings to Tunnel Protocols	16
7.1. Embedding LOOPS in Geneve	17
8. IANA Considerations	17
9. Security Considerations	17
9.1. Threat model	17
9.2. Discussion	18
10. References	18
10.1. Normative References	18
10.2. Informative References	19
Appendix A. Protocol used in Prototype Implementation	21
A.1. Block Code FEC	22
Appendix B. Transparent mode	22

B.1. Packet identification	24
B.2. Generic information and protocol operation	25
B.3. A hybrid mode	25
Acknowledgements	27
Authors' Addresses	27

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].

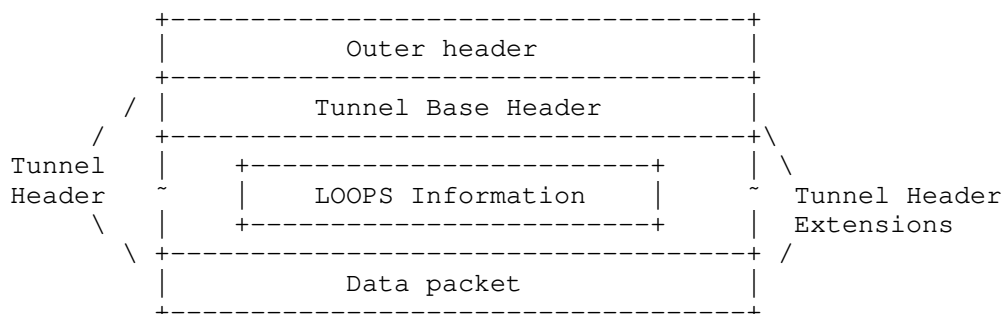


Figure 1: Packet in Tunnel with LOOPS Information

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.

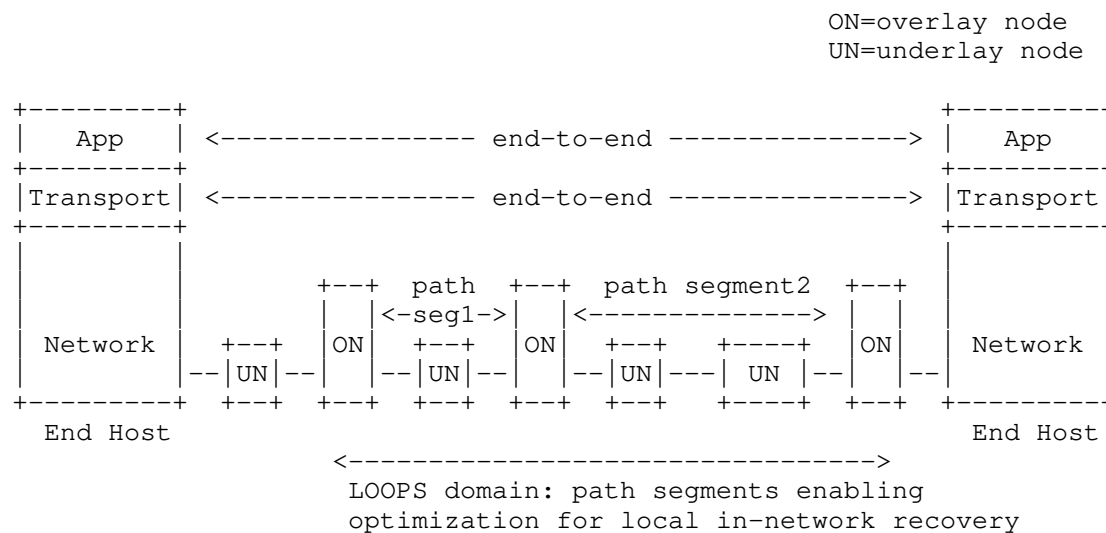


Figure 2: LOOPS Usage Scenario

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 $(\text{Acked-PSN} \& \sim 7) + 8 * (\text{delta} + 1)$ 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., Dukkupati, 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.

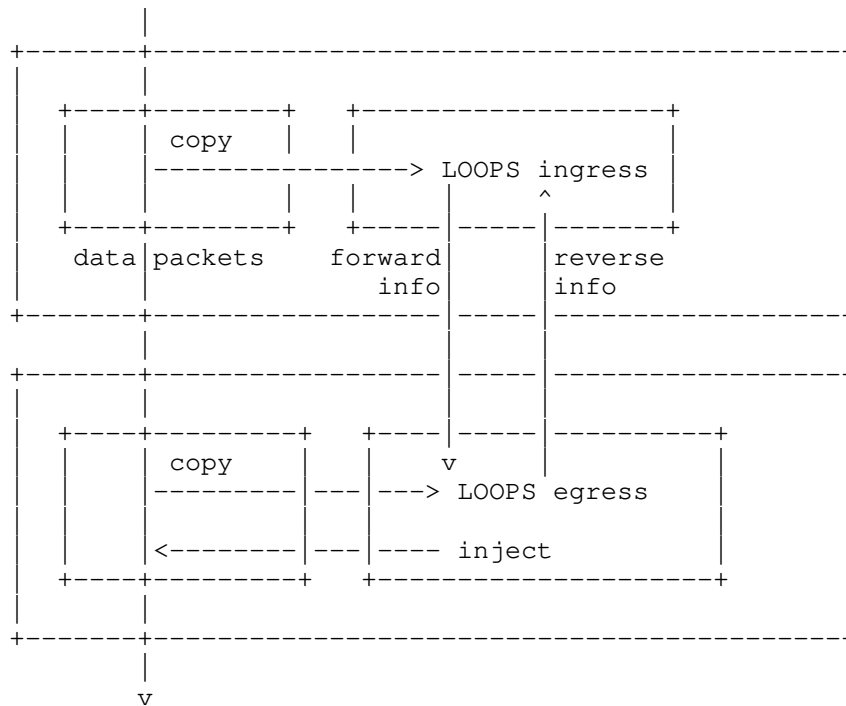


Figure 3: LOOPS Transparent Mode

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:

```

      0                               1                               2                               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|0|0|0|1|         000000000         | 000 |         Protocol Type         |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Sequence Number                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

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 $8 * (\text{end-psn-delta} + 1)$ 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

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: May 20, 2021

S. Yang
CUHK (SZ)
X. Huang
R.W. Yeung
CUHK
J.K. Zao
NCTU

November 16, 2020

BATS Coding Scheme for Multi-hop Data Transport
draft-yang-nwcrg-bats-04

Abstract

This document describes a BATS coding scheme for communication through multi-hop networks. BATS code is a class of efficient linear network coding scheme with a matrix generalization of fountain codes as the outer code, and batch-based linear network coding as the inner code.

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 May 20, 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	2
1.1. Requirements Language	3
2. Procedures	3
2.1. Introduction	4
2.2. Data Delivery Procedures	5
2.2.1. Source Node Data Partitioning and Padding	5
2.2.2. Source Node Outer Code Encoding Procedure	6
2.2.3. Recoding Procedures	7
2.2.4. Destination Node Procedures	8
2.3. Recommendation for the Parameters	8
2.4. Example DDP Packet Format	9
2.4.1. Packet Header	9
2.4.2. Packet Payload	10
2.4.3. Packet Footer	10
3. BATS Code Specification	11
3.1. Common Parts	11
3.2. Outer Code Encoder	12
3.3. Inner Code Encoder (Recoder)	13
3.4. Belief Propagation Decoder	13
4. IANA Considerations	14
5. Security Considerations	14
5.1. Provision of Confidentiality Protection	14
5.2. Countermeasures against Pollution Attacks	15
6. Data Delivery Protocol Considerations	16
7. References	16
7.1. Normative References	16
7.2. Informative References	17
Appendix A. Additional Stuff	17
Authors' Addresses	17

1. Introduction

This document specifies a BATS code [BATS] scheme for data delivery in multi-hop networks. The BATS code described here includes an outer code and an inner code. The outer code is a matrix generalization of fountain codes (see also the RapterQ code described in RFC 6330 [RFC6330]), which inherits the advantages of reliability and efficiency and possesses the extra desirable property of being network coding compatible. The inner code, also called recoding, is formed by linear network coding for combating packet loss, improving

the multicast efficiency, etc. A detailed design and analysis of BATS codes are provided in the BATS monograph [BATSMonograph].

The BATS coding scheme can be applied in multi-hop networks formed by wireless communication links, which are inherently unreliable due to interference. Existing transport protocols like TCP use end-to-end retransmission, while network protocols such as IP might enable store-and-forward at the relays, so that packet loss would accumulate along the way.

The BATS coding scheme can be used for various data delivery applications like file transmission, video streaming over wireless multi-hop networks, etc. Different from traditional forward error correcting (FEC) schemes that are applied either hop-by-hop or end-to-end, the BATS coding scheme combines the end-to-end coding (the outer code) with certain hop-by-hop coding (the inner code), and hence can potentially achieve better performance.

The coding scheme described here can be used in a network with multiple communication flows. For each flow, the source node encodes the data for transmission separately. Inside the network, however, it is possible to mix the packets from different flows for recoding. In this document, we describe a simple case where recoding is performed within each flow. Note that the same encoding/decoding scheme described here can be used with different recoding schemes as long as they follow the principle as we illustrate in this document.

The purpose of this document is a preliminary BATS coding scheme for researchers and engineers so that they can develop further the network communication applications/protocols based on BATS codes. Towards a sophisticated BATS code based network protocol, several important research directions should be considered. One is about the security issues. Another important current research is to design the corresponding congestion control and routing algorithms for BATS codes.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Procedures

2.1. Introduction

A BATS coding scheme includes an outer code encoder (also called encoder), an inner code encoder (also called recoder) and a decoder. The BATS coding scheme can be used for a single data flow that includes a single source and one or multiple destinations. Thus there exists only one encoder with multiple recoders and decoders. The BATS coding scheme described in this document can be used by a Data Delivery Protocol (DDP) with the following procedures.

Outer Code Encoding at a source node which has the data for transmission:

- * The DDP provides the data to be delivered and the related information to the BATS encoder.
- * The BATS encoder generates a sequence of batches, each consisting of a set of coded packets and the information pertaining to the batch.

The batches generated at the source node are further recoded before transmitting:

- * A BATS recoder generates recoded packets of a batch.
- * The DDP forms and transmits the DDP packets using the batches and the corresponding batch information.

Recoding at an intermediate node that does not need the data:

- * The DDP extracts the batches and the corresponding batch information from its received DDP packets.
- * A BATS recoder generates recoded packets of a batch.
- * The DDP forms and transmits DDP packets using the recoded packets and the corresponding batch information.

Decoding at a destination node that needs the data:

- * The DDP extracts the batches and the corresponding batch information from its received DDP packets.
- * A BATS decoder tries to recover the transmitted data using the received batches.
- * The DDP sends the decoded data to the application that needs the data.

2.2. Data Delivery Procedures

Suppose that the DDP has F octets of data for transmission. We describe the procedures of one BATS session for transmitting the F octets. There is a limit on F of a single BATS session. If the total data has more than the limit, the data needs to be transmitted using multiple BATS sessions. The limit on F of a single BATS session depends on the MTU (maximum transmission unit) of the network, which MUST be known by the DDP. We have F is no more than $(MTU-10)2^{16-1}$ octets.

2.2.1. Source Node Data Partitioning and Padding

The DDP first determines the following parameters:

- o Batch size (M): the number of coded packets in a batch.
- o Recoding field size (q): the number of elements in the finite field for recoding. q is 2 or 2^8
- o BATS payload size (T_0): the number of payload octets in a BATS packet, including the coded data and the coefficient vector.

Based on the above parameters, the parameters T , O and K are calculated as follows:

- o O : the number of octets of a coefficient vector, calculated as $O = \text{ceil}(M \cdot \log_2(q)/8)$.
- o T : the number of data octets of a BATS packet, calculated as $T = T_0 - O$.
- o K : number of source packets, calculated as $K = \text{floor}(F/T) + 1$.

The data MUST be padded to have $T \cdot K$ octets, which will be partitioned into K source packets $b[0], \dots, b[K-1]$, each of T octets. In our padding scheme, $b[0], \dots, b[K-2]$ are filled with data bits, and $b[K-1]$ is filled with the remaining data octets and padding octets. Let $P = K \cdot T - F$ denote the number of padding octets. We use $b[K-1, 0], \dots, b[K-1, T-P-1]$ to denote the $T-P$ source octets and $b[K-1, T-P], \dots, b[K-1, T-1]$ to denote the P padding octets in $b[K-1]$, respectively. The padding process is shown in Figure 1.


```

Z = T - P
Let bl be the last source packet b[K-1]
for i = 1, 2, ... do
  if Z + i >= T - 1 do
    bl[Z...T-1] = i
    break
  bl[Z...Z+i-1] = i
  Z = Z + i

```

Figure 1: Data Padding Process

2.2.2. Source Node Outer Code Encoding Procedure

The DDP provides the BATS encoder with the following information:

- o Batch size (M): the number of coded packets in a batch.
- o Recoding field size (q): the number of elements in the finite field for recoding.
- o MAX_DEG: the size of DD.
- o The degree distribution (DD), which is an unsigned integer array of size MAX_DEG+1.
- o A sequence of batch IDs (j , $j = 0, 1, \dots$).
- o Number of source packets (K).
- o Packet size (T): the number of octets in a source packet.
- o The source packets ($b[i]$, $i = 0, 1, \dots, K-1$).

Using this information, the (outer code) encoder generates a batch for each batch ID. For the batch ID j , the encoder returns the DDP that contains

- o a sparse degree $d[j]$, and
- o M coded packets ($x[j,i]$, $i = 0, 1, \dots, M-1$), each containing TO octets.

The DDP will use the batches to form DDP packets to be transmitted to other network nodes towards the destination nodes. The DDP MUST deliver with each coded packet its

- o d: sparse degree

- o BID: batch ID

The DDP MUST deliver the following information to each recoder:

- o M: batch size M
- o q: recoding field size

The DDP MUST deliver the following information to each decoder:

- o M: batch size
- o q: recoding field size
- o K: the number of source packet
- o T: the number of octets in a source packet

The BID is used by both recoders and decoders. The BATS payload size TO MUST be known by all the nodes.

The DDP will also include some necessary extra information in the packet header so that the network nodes can identify different BATS sessions, and different end-to-end communication flows. However, such specifications are beyond the scope of this document.

2.2.3. Recoding Procedures

Both the source node and the intermediate nodes perform recoding on the batches before transmission. At the source node, the recoder receives the batches from the outer code encoding procedure. At an intermediate node, the DDP receives the DDP packets from the other network nodes, and should be able to extract coded packets and the corresponding batch information from these packets.

The DDP provides the recoder with the following information:

- o the batch size M,
- o the recoding field size q,
- o a number of received coded packets of the same batch, each containing TO octets, and
- o link statistics, e.g., packet loss rates.

For a received batch, the recoder determines a positive integer M_r , the number of recoded packets to be transmitted for the batch. The

recoder uses the information provided by the DDP to generate Mr recoded packets, each containing TO octets. The DDP uses the Mr recoded packets to form the DDP packets for transmitting.

2.2.4. Destination Node Procedures

A destination node needs the data transmitted by the source node. At the destination node, the DDP receives DDP packets from the other network nodes, and should be able to extract coded packets and the corresponding batch information from these packets.

The DDP provides the decoder with the following information:

- o M: batch size,
- o q: recoding field size,
- o K: the number of source packets
- o T: the number of octets of a source packet
- o A sequence of batches, each of which is formed by a number of coded packets belonging to the same batch, with their corresponding batch IDs and degrees.

The decoder uses this information to decode the K source packets. If successful, the decoder returns the recovered K source packets to the DDP, which will use the K source packets to form the F octets data. The recommended padding process is shown as follows:

```
// this procedure returns the number P of padding octets
// at the end of b[K-1]
Let bl be the last decoded source packet b[K-1]
PL = bl[T-1]
if PL == 1 do
    return P = 1
WI = T - 1
while bl[WI] == PL do
    WI = WI - 1
return P = (1 + bl[WI]) * bl[WI] + T - WI - 1
```

Figure 2: Data Depadding Process

2.3. Recommendation for the Parameters

The recommendation for the parameters M and q is shown as follows:

- o When q=2, M=16,32,64

- o When $q=256$, $M=8,16,32,64$

It is RECOMMENDED that K is at least 128. However, the encoder/decoder SHALL support an arbitrary positive integer value less than 2^{16} .

2.4. Example DDP Packet Format

A DDP can form a DDP packet with a header (5 octets), a footer (3 octets) and a payload (T_O octets). A DDP packet has totally $8+T_O$ octets.

2.4.1. Packet Header

The BATS packet header has 40 bits (5 octets) and includes fields Packet_Count, M_q , Batch_ID, and Degree.

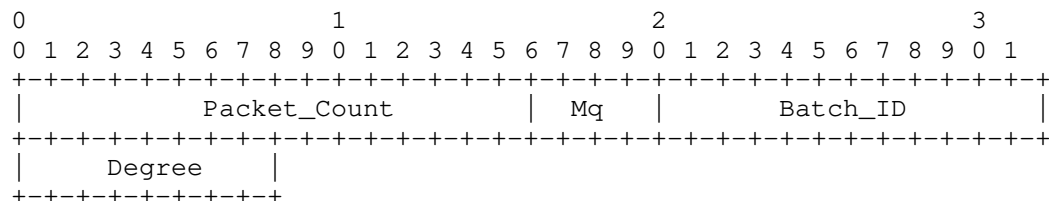


Figure 3: BATS packet header format.

- o Packet_Count: 16-bit unsigned integer, specifying the number K of packets of the BATS session.
- o M_q : 4-bit unsigned integer to specify the value of M and q as Table 1.
- o Batch_ID: 12-bit unsigned integer, specifying the batch ID BID of the batch the packet belonging to.
- o Degree: 8-bit unsigned integer, specifying the batch degree d of the batch the packet belonging to.

Mq	M	q	O
0010	16	2	2
0100	32	2	4
0110	64	2	8
0001	8	256	8
0011	16	256	16
0101	32	256	32
0111	64	256	64

Table 1: Values of Mq field

2.4.2. Packet Payload



Figure 4: BATS packet payload format.

The payload has TO octets, where the first O octets contain the coefficient vector and the remaining T octets contain the coded data. Information in both fields MAY be encoded in JSON (ASCII) or protobuf (binary) formats.

- o coefficient vector: O octets. The range of the value of O is in Table 1.
- o coded data: T octets. T is at most MTU - 10, where 10 is the total of the header and footer length plus the minimum value of O.

2.4.3. Packet Footer

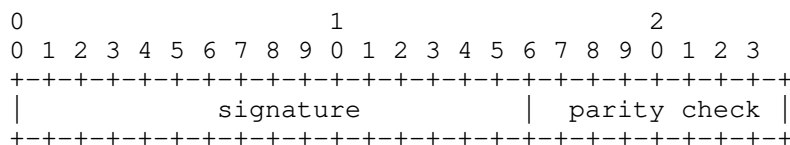


Figure 5: BATS packet footer format.

The footer has three octets.

- o signature: 2 octets. A signature of the individual packet to prevent pollution attack.

- o parity check: 1 octet. A parity check field used to verify the correctness of the packet.

3. BATS Code Specification

3.1. Common Parts

The T octets of a source packets are treated as a column vector of T elements in $GF(256)$. Linear algebra and matrix operations over finite fields are assumed in this section.

Suppose that a pseudorandom number generator `Rand()` which generates an unsigned integer of 32 bits is shared by both encoding and decoding. The pseudorandom generator can be initialized by `Rand_Init(S)` with seed S . When S is not provided, the pseudorandom generator is initialized arbitrarily. One example of such a pseudorandom generator is defined in RFC 8682 [RFC8682].

A function called `BatchSampler` is used in both encoding and decoding. The function takes two integers j and d as input, and generates an array `idx` of d integers and a $d \times M$ matrix G . The function first initializes the pseudorandom generator with j , sample d distinct integers from 0 to $K-1$ as `idx`, and sample $d \times M$ integers from 0 to 255 as G . See the pseudocode in Figure 6.

```
function BatchSampler(j,d)
    // initialize the pseudorandom generator by seed j.
    Rand_Init(j)
    // sample d distinct integers between 0 and K-1.
    for k = 0, ..., d-1 do
        r = Rand() % K
        while r already exists in idx do
            r = Rand() % K
        idx[k] = r

    // sample d x M matrix
    for r = 0, ..., d-1 do
        for c = 0, ..., M-1 do
            G[r,c] = Rand() % 256

    return idx, G
```

Figure 6: Batch Sampler Function

3.2. Outer Code Encoder

Define a function called DegreeSampler that return an integer d using the degree distribution DD . We expect that the empirical distribution of the returning d converges to $DD(d)$ when $d < K$. One design of DegreeSampler is illustrated in Figure 7.

```
function DegreeSampler(j, DD)
  Let CDF be an array
  CDF[0] = 0
  for i = 1, ..., MAX_DEG do
    CDF[i] = CDF[i-1] + DD[i]
  Rand_Init()
  r = Rand() % CDF[MAX_DEG]
  for d = 1, ..., MAX_DEG do
    if r >= CDF[d] do
      return min(d,K)
  return min(MAX_DEG,K)
```

Figure 7: Degree Sampler Function

Let $b[0], b[1], \dots, b[K-1]$ be the K source packets. A batch with BID j is generated using the following steps.

- o Obtain a degree d by calling DegreeSampler with input j .
- o Obtain idx and $G[j]$ by calling BatchSampler with input j and d .
- o Let $B[j] = (b[idx[0]], b[idx[1]], \dots, b[idx[d-1]])$. Form the batch $X[j] = B[j]*G[j]$, whose dimension is $T \times M$.
- o Form the $TO \times M$ matrix $Xr[j]$, where the first O rows of $Xr[j]$ form the $M \times M$ identity matrix I with entries in $GF(q)$, and the last T rows of $Xr[j]$ is $X[j]$.

See the pseudocode of the batch generating process in Figure 8.

```
function GenBatch(j)
  d = DegreeSampler(j)
  (idx, G) = BatchSampler(j,d)
  B = (b[idx[0]], b[idx[i]], ..., b[idx[d-1]])
  X = B * G
  Xr = [I_M; X]
  return Xr
```

Figure 8: Batch Generation Function

3.3. Inner Code Encoder (Recoder)

The inner code comprises (random) linear network coding applied on the coded packets belonging to the same batch. At a particular network node, recoded packets are generated by (random) linear combinations of the received coded packets of a batch. The recoded packets have the same BID, sparse degree and coded packet length.

The number M_r of recoded packets for a batch is decided first by the recoder. M_r can be set as M . When the link statistics is known, the recoder can try to obtain the link packet loss rate e for the link to transmit the recoded batch, and set M_r to be $(1+e)M$.

Suppose that coded packets $xr[i]$, $i = 0, 1, \dots, r-1$, which have the same BID j , have been received at an intermediate node. Using the recommended packet format, it can be verified whether the corresponding packet headers of these coded packets are the same. Then a recoded packet can be generated by one of the following two approaches:

- o forwarding: when receiving $xr[i]$, directly use $xr[i]$ as a recoded packet.
- o linear combination recoding: (randomly) choose a sequence of coefficients $c[i]$, $i = 0, 1, \dots, r-1$ from $GF(q)$. Generate $c[0]xr[0]+c[1]xr[1]+\dots+c[r-1]xr[r-1]$ as a recoded packet.

A recoder can combine these two approaches to generate recoded packets. For example, the recoder will output $xr[i]$, $i = 0, 1, \dots, r-1$ as r systematic recoded packets and generate M_r-r recoded packets using linear combinations of randomly chosen coefficients.

3.4. Belief Propagation Decoder

The decoder receives a sequence of batches $Yr[j]$, $j = 0, 1, \dots, n-1$, each of which is a TO-row matrix over $GF(256)$. The degree $d[j]$ of batch j is also known. Let $Y[j]$ be the submatrix of the last T rows of $Yr[j]$. When $q = 256$, let $H[j]$ be the first M rows of $Yr[j]$; when $q = 2$, let $H[j]$ be the matrix over $GF(256)$ formed by embedding each bit in the first $M/8$ rows of $Yr[j]$ into $GF(256)$.

By calling BatchSampler with input j and $d[j]$, we obtain $idx[j]$ and $G[j]$. According to the encoding and recoding processes described in Section 3.2 and Section 3.3, we have the system of linear equations $Y[j] = B[j]G[j]H[j]$ for each received batch with ID j , where $B[j] = (b[idx[j],0], b[idx[j],1], \dots, b[idx[j],d-1])$ is unknown.

We describe a belief propagation (BP) decoder that can efficiently solve the source packets when a sufficient number of batches have been received. A batch j is said to be decodable if $\text{rank}(G[j]H[j]) = d[j]$ (i.e., the system of linear equations $Y[j] = B[j]G[j]H[j]$ with $B[j]$ as the variable matrix has a unique solution). The BP decoding algorithm has multiple iterations. Each iteration is formed by the following steps:

- o Decoding step: Find a batches j that is decodable. Solve the corresponding system of linear equations $Y[j] = B[j]G[j]H[j]$ and decode $B[j]$.
- o Substitution step: Substitute the decoded source packets into undecodable batches. Suppose that a decoded source packet $b[k]$ is used in generating a undecodable $Y[j]$. The substitution involves 1) removing the entry in $\text{idx}[j]$ corresponding to k , 2) removing the row in $G[j]$ corresponding to $b[k]$, and 3) reducing $d[j]$ by 1.

The BP decoder repeats the above steps until no batches are decodable during the decoding step.

4. IANA Considerations

This memo includes no request to IANA.

5. Security Considerations

Subsuming both Random Linear Network Codes (RLNC) and fountain codes, BATS codes naturally inherit both their desirable capability of offering confidentiality protection as well as their vulnerability towards pollution attacks.

5.1. Provision of Confidentiality Protection

Since the transported messages are linearly combined with random coefficients at each recoding node, it is statistically impossible to recover the individual messages by capturing the coded messages at any one or small number of nodes. As long as the coding matrices of the transported messages cannot be fully recovered, any attempt of decoding any particular symbol of the transported messages is equivalent to random guessing [Bhattad05].

The threat towards confidentiality, however, also exists in the form of eavesdropping on the initial encoding process, which takes place at the encoding nodes. In these nodes, the transported data are presented in plain text and can be read along their transfer paths. Hence, information isolation between the encoding process and all other user processes running on the node must be assured.

In addition, the authenticity and trustworthiness of the encoding, recoding and decoding program running on all the nodes must be attested by a trusted authority. Such a measure is also necessary in countering pollution attacks.

5.2. Countermeasures against Pollution Attacks

Like all network codes, BATS codes are vulnerable under pollution attacks. In these attacks, one or more compromised coding node(s) can pollute the coded messages by injecting forged messages into the coding network and thus prevent the receivers from recovering the transported data correctly.

The research community has long been investigating the use of various signature schemes (including homomorphic signatures) to identify the forged messages and stall the attacks (see [Zhao07], [Yu08], [Agrawal09]). However, these countermeasures are regarded as being too computationally expensive to be employed in broadband communications. Hence, a system-level approach based on Trusted Computing [TC-Wikipedia] is proposed as a practical alternative to protect BATS codes against pollution attacks. This Trusted Computing based protection consists of the following countermeasures:

1. Attestation and Validation of all BATS encoding, recoding and decoding nodes in the network. Remote attestation and repetitive validation of the identity and capability of these node based on valid public key certificates with proper authorization MUST be a pre-requisite for admitting these nodes to a network and permitting them to remain on that network.
2. Attestation of all encoding, recoding and decoding programs used in the coding nodes. All programs used to perform the BATS encoding, recoding and decoding processes MUST be remotely attested before they are permitted to run on any of the coding nodes. Reloading or alteration of programs MUST NOT be permitted during an encoding session. Programs MUST be attested or validated again when they are executed in new execution environments instantiated even in the same node.
3. Original Authentication of all coded messages using network level security protocols such as IPsec or Peer Authentication over session-based communication using transport level security protocols such as TLS/DTLS MUST be employed in order to provide Message Origin or Communication Peer Authentication to every coded message sent through the coding network.

6. Data Delivery Protocol Considerations

In addition to the security consideration, there are other issues to be considered towards a fully functionally DDP based on the BATS coding scheme described in this document. Here we discuss the related routing and congestion control considerations, which are research issues that need further elaborations. The general information theoretical guideline is as follows: The outer code of a BATS code can be regarded as a channel code for the channel induced by the inner code, and hence the routing and congestion control algorithms should try to maximize the capacity of the channel induced by the inner code.

The BATS coding scheme is flexible for implementing multipath data transmission. Different batches can be transmitted on a different path between a source node and a destination node. For multicast, however, we may allow the combination of certain batches during recoding to improve the multicast throughput. Moreover, to benefit from multiple source nodes, we would need different source node generates statistically independent batches.

Congestion control for a DDP employing a BATS code scheme has several extra design considerations. First, the end-to-end throughput should be measured by the average rank rate (the total rank of all the received batches over the time). Second, both the rate of transmitting batches at the source nodes and the number of recoded packets generated by recoding should be adjusted for congestion control. Note that the number of recoded packets can be different for different batch and at different node. Last but not the least, for scenarios like wireless multihop networks, congestion control needs to be performed hop-by-hop, instead of end-to-end as in TCP, due to the high packet loss rate and the long end-to-end delay.

7. References

7.1. 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>>.
- [RFC8682] Saito, M., Matsumoto, M., Roca, V., Ed., and E. Baccelli, "TinyMT32 Pseudorandom Number Generator (PRNG)", RFC 8682, DOI 10.17487/RFC8682, January 2020, <<https://www.rfc-editor.org/info/rfc8682>>.

7.2. Informative References

- [Agrawal09] Agrawal, S. and D. Boneh, "Homomorphic MACs: MAC-based integrity for network coding", International Conference on Applied Cryptography and Network Security , 2009.
- [BATS] Yang, S. and R. Yeung, "Batched Sparse Codes", IEEE Transactions on Information Theory 60(9), 5322–5346, 2014.
- [BATSMonograph] Yang, S. and R. Yeung, "BATS Codes: Theory and Practice", Morgan & Claypool Publishers , 2017.
- [Bhattad05] Bhattad, K. and K. Narayanan, "Weakly Secure Network Coding", ISIT , 2007.
- [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>>.
- [TC-Wikipedia] "Trusted Computing",
Wikipedia https://en.wikipedia.org/wiki/Trusted_Computing.
- [Yu08] Yu, Z., Wei, Y., Ramkumar, B., and Y. Guan, "An Efficient Signature-Based Scheme for Securing Network Coding Against Pollution Attacks", INFOCOM , 2008.
- [Zhao07] Zhao, F., Kalker, T., Medard, M., and K. Han, "Signatures for content distribution with network coding", ISIT , 2007.

Appendix A. Additional Stuff

This becomes an Appendix.

Authors' Addresses

Shenghao Yang
CUHK(SZ)
Shenzhen, Guangdong
China

Phone: +86 755 8427 3827
Email: shyang@cuhk.edu.cn

Xuan Huang
CUHK
Hong Kong, Hong Kong SAR
China

Phone: +852 3943 8375
Email: 1155136647@link.cuhk.edu.hk

Raymond W. Yeung
CUHK
Hong Kong, Hong Kong SAR
China

Phone: +852 3943 8375
Email: whyeung@ie.cuhk.edu.hk

John K. Zao
NCTU
Hsinchu, Taiwan
China

Email: jkzao@ieee.org