

NWCRG
Internet-Draft
Intended status: Informational
Expires: May 3, 2021

N. Kuhn
CNES
E. Lochin
ENAC
F. Michel
UCLouvain
M. Welzl
University of Oslo
October 30, 2020

Coding and congestion control in transport
draft-irtf-nwcrg-coding-and-congestion-04

Abstract

Forward Erasure Correction (FEC) is a reliability mechanism that is distinct and separate from the retransmission logic in reliable transfer protocols such as TCP. Using FEC coding can help deal with transfer tail losses or with networks having non-congestion losses. However, FEC coding mechanisms should not hide congestion signals. This memo offers a discussion of how FEC coding and congestion control can coexist. Another objective is to encourage the research community to also consider congestion control aspects when proposing and comparing FEC coding solutions in communication systems.

This document is the product of the Coding for Efficient Network Communications Research Group (NWCRG). The scope of the document is end-to-end communications: FEC coding for tunnels is out-of-the scope of the document.

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	3
2. Separate channels, separate entities	4
3. FEC above the transport	6
3.1. Flowchart	6
3.2. Discussion	7
4. FEC within the transport	8
4.1. Flowchart	8
4.2. Discussion	8
5. FEC below the transport	9
5.1. Flowchart	9
5.2. Discussion	9
6. Fairness, redundacy rate and congestion signals	10
6.1. Fairness, a policy concern	10
6.2. Fairness and impact on non-coded flows	11
6.2.1. FEC above the transport	11
6.2.2. FEC within the transport	11
6.2.3. FEC below the transport	11
6.3. Congestion control and recovered symbols	11
6.3.1. FEC above the transport	11
6.3.2. FEC within the transport	12
6.3.3. FEC below the transport	12
6.4. Interactions between congestion control and coding rates	12
6.4.1. FEC above the transport	12
6.4.2. FEC within the transport	12
6.4.3. FEC below the transport	13
6.5. On the useless repair symbols	13
6.5.1. FEC above the transport	13
6.5.2. FEC within the transport	13
6.5.3. FEC below the transport	13
7. Open research questions	13
7.1. Activities related to congestion control and coding	13

7.2. Open research questions	14
7.3. Advices for evaluating coding mechanisms	14
8. Acknowledgements	15
9. IANA Considerations	15
10. Security Considerations	15
11. Informative References	15
Authors' Addresses	17

1. Introduction

There are cases where deploying FEC coding improves the performance of a transmission. As an example, it may take time for the sender to detect transfer tail losses (losses that occur at the end of a transfer, where e.g., TCP obtains no more ACKs to repair the loss via retransmission quickly). Allowing the receiver to recover such losses instead of having to rely on a retransmission could improve the experience of applications using short flows. Another example are networks where non-congestion losses are persistent and prevent a sender from exploiting the link capacity.

Coding is a reliability mechanism that is distinct and separate from the loss detection of congestion controls. [RFC5681] defines TCP as a loss-based congestion control; since FEC coding repairs such losses, blindly applying it may easily lead to an implementation that also hides a congestion signal from the sender. It is important to ensure that such information hiding does not occur.

FEC coding and congestion control can be seen as two separate channels. In practice, implementations may mix the signals that are exchanged on these channels. This memo offers a discussion of how FEC coding and congestion control can coexist. Another objective is to encourage the research community also to consider congestion control aspects when proposing and comparing FEC coding solutions in communication systems. This document does not aim at proposing guidelines for characterizing FEC coding solutions.

The proposed document considers an end-to-end unicast data transfer with FEC coding at the application (above the transport), within the transport or directly below the transport. The typical application scenario considered in the current version of the document is a client browsing the web or watching a live video. This memo may be extended to cases with multiple paths.

This document represents the collaborative work and consensus of the Coding for Efficient Network Communications Research Group (NWCRG); it is not an IETF product and is not a standard. The document follows the terminology proposed in the taxonomy document [RFC8406].

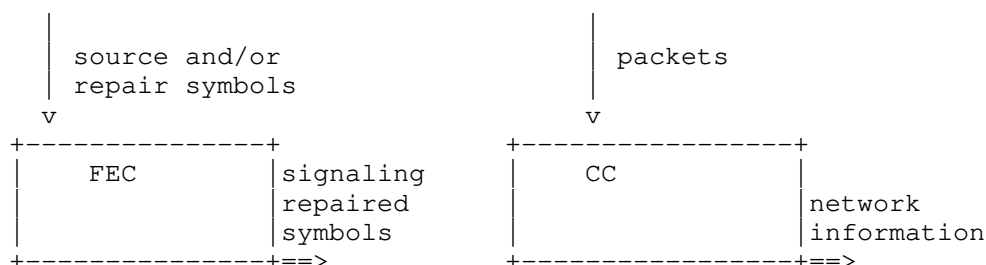


Figure 3: Separate entities (receiver-side)

Figure 2 and Figure 3 provide more details than Figure 1. Some elements are introduced:

- o 'network information' (input control plane for the transport including CC): refers not only to the network information that is explicitly signaled from the receiver, but all the information a congestion control obtains from a network (e.g., TCP can estimate the latency and the available capacity at the bottleneck).
- o 'requirements' (input control plane for the transport including CC): refers to application requirements such as upper/lower rate bounds, periods of quiescence, or a priority.
- o 'sending rate (or window)' (output control plane for the transport including CC): refers to the rate at which a congestion control decides to transmit packets based on 'network information'.
- o 'signaling repaired symbols' (input control plane for the FEC): refers to the information a FEC sender can obtain from a FEC receiver about the performance of the FEC solution as seen by the receiver.
- o 'coding rate' (output control plane for the FEC): refers to the coding rate that is used by the FEC solution.
- o 'source and/or repair symbols' (data plane for both the FEC and the CC): refers to the data that is transmitted. The sender can decide to send source symbols only (meaning that the coding rate is 0), repair symbols only (if the solution decides not to send the original source packets) or a mix of both.

The inputs to FEC (incoming data packets without repair symbols, and signaling from the receiver about losses and/or repaired symbols) are distinct from the inputs to CC. The latter calculates a sending rate or window from network information, and it takes the packet to send as input, sometimes along with application requirements such as

upper/lower rate bounds, periods of quiescence, or a priority. It is not clear that the ACK signals feeding into a congestion control algorithm are useful to FEC in their raw form, and vice versa - information about repaired blocks may be quite irrelevant to a CC algorithm.

The choice of the adequate transport layer may be related to application requirements:

- o In the case of an unreliable data transfer, the transport layer may provide a non-reliable transport service (e.g. UDP or DCCP [RFC4340] or a partially reliable transport protocol such as SCTP with partial reliability [RFC3758]). Depending on the amount of redundancy and network conditions, there could be cases where it becomes impossible to carry traffic.
- o In the case of a reliable data transfer, the transport layer may implement a retransmission mechanism to guarantee the reliability of the file transfer (e.g. TCP). Depending on how the FEC and CC functions are scheduled (FEC above CC, FEC in CC, FEC below CC), the impact of reliable transport on the FEC reliability mechanisms is different.

3. FEC above the transport

3.1. Flowchart

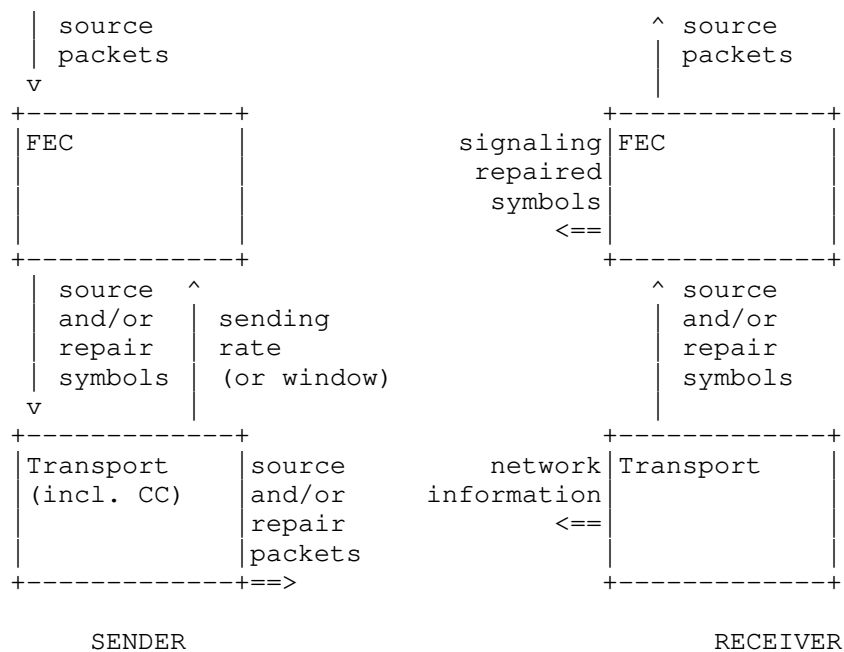


Figure 4: FEC above the transport

Figure 4 present an architecture where FEC operates on top of the transport.

3.2. Discussion

The advantage of this approach is that the FEC overhead does not contribute to congestion in the network. When congestion control is implemented at the transport layer, the repair symbols are sent following the congestion window. This approach can result in improved quality of experience for latency sensitive applications such as VoIP.

This approach requires that the transport protocol does not implement a fully reliable data transfer service (e.g., based on lost packet retransmission). UDP is an example of a protocol for which this approach is relevant. For reliable transfers, coding usage does not guarantee better performance and would mainly reduce goodput for large file transfers.

This discussion section is extended in Section 6.

4. FEC within the transport

4.1. Flowchart

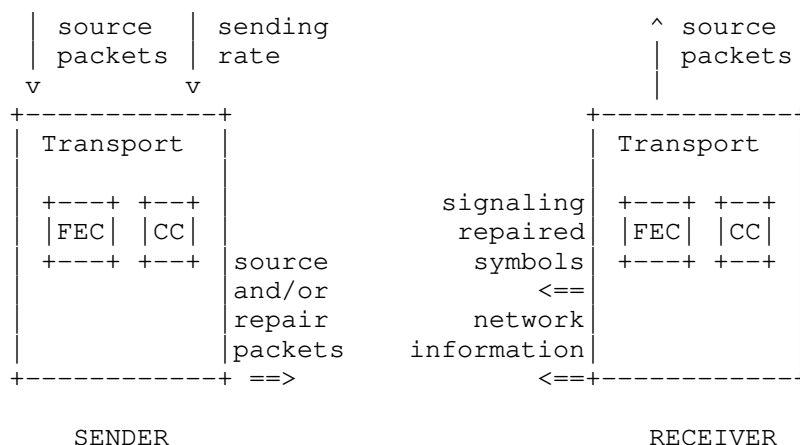


Figure 5: FEC in the transport

Figure 5 presents an architecture where FEC operates within the transport. The repair symbols are sent within what the congestion window allows, such as in [CTCP].

4.2. Discussion

The advantage of this approach allows a joint optimization between the CC and the FEC. Moreover, the transmission of repair symbols does not add congestion in potentially congested networks but helps repair lost packets (such as tail losses).

For reliable transfers, including redundancy reduces goodput for large file transfers but the amount of repair symbols can be adapted, e.g. depending on the congestion window size. There is a trade-off between the cost in capacity used to transmit source packets and the benefits brought out by transmitting repair symbols (e.g. unlocking the receive buffer if this is limiting). The coding ratio needs to be carefully designed. For small files, sending repair symbols when there is no more data to transmit could help to reduce the transfer time. In general, sending repair symbols could avoid a silent period between the transmission of the last packet in the send buffer and 1) firing the retransmission of lost packets, or 2) the transmission of new packets.

This discussion section is extended in Section 6.

5. FEC below the transport

5.1. Flowchart

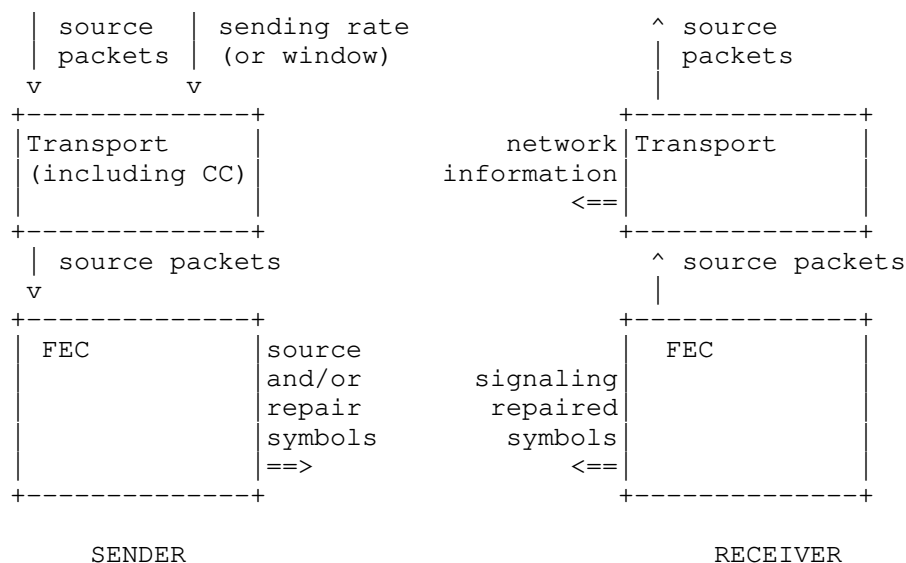


Figure 6: FEC below the transport

Figure 6 presents an architecture where FEC is applied end-to-end below the transport layer, but above the link layer. Note that it is common to apply FEC at the link layer, in which it contributes to the total capacity that a link exposes to upper layers. This application of FEC is out of scope of this document. In the scenario considered here, the repair symbols are sent on top of what is allowed by the congestion control.

5.2. Discussion

In this case, including redundancy adds congestion without reducing goodput but leads to potential fairness issues. The effective bitrate is indeed higher than the CC's computed fair share due to the sending of repair symbols and the losses are hidden from the transport. This may cause a problem for loss-based congestion detection, but it is not a problem for delay-based congestion detection.

The advantage of this approach is that it can result in performance gains when there are persistent transmission losses along the path.

The drawback of this approach is that it can induce congestion in already congested networks. The coding ratio needs to be carefully designed.

Examples of the solution could be adding a given percentage of the congestion window as supplementary symbols or sending a given amount of repair symbols at a given rate. The redundancy flow can be decorrelated from the congestion control that manages source packets: a separate congestion control entity could be introduced to manage the amount of repaired packets to transmit on the FEC channel. The separate congestion control instances could be made to work together while adhering to priorities, as in coupled congestion control for RTP media [RFC8699] in case all traffic can be assumed to take the same path, or otherwise with a multipath congestion window coupling mechanism as in Multipath TCP [RFC6356]. Another possibility would be to exploit a lower than best-effort congestion control [RFC6297] for repair symbols.

This discussion section is extended in Section 6.

6. Fairness, redundancy rate and congestion signals

The objective of this section is to further detail some aspects that have been expressed in previous discussion subsections.

6.1. Fairness, a policy concern

The contract between the client and the operator may guarantee a minimum data-rate (e.g. mobile networks). However, for residential accesses, the data-rate can be guaranteed for the customer premises equipment, but not necessarily for the client. The quality of service that guarantees fairness between the different clients can be seen as a policy concern [I-D.briscoe-tsvarea-fair].

While flow level fairness does not embody the actual application level fairness, the share of available capacity between single flows can help assess when one flow starves the other. Clients may share a bottleneck that may not be ruled by a quality of service mechanism, e.g. in case of:

- o a mobile network client running several applications;
- o two clients on a residential access.

This document considers fairness as an index to quantify the impact of the addition of coded flows on non-coded flows when they share the same bottleneck. This document does not aim at contributing to the

definition of fairness at a wider scale. This document assumes that the non-coded flows respond to congestion signals from the network.

6.2. Fairness and impact on non-coded flows

6.2.1. FEC above the transport

The addition of coding within the flow does not impact on the interaction between coded and non-coded flows. This interaction would mainly depend on the congestion controls embedded in each host.

6.2.2. FEC within the transport

The addition of coding within the flow may impact the congestion control mechanism and hide congestion losses. Specific interaction between congestion controls and coding schemes can be proposed (see Section 6.3, Section 6.4 and Section 6.5). If no specific interaction is introduced, the coding scheme may hide congestion losses from the congestion controller and the description of Section 6.2.3 may apply.

6.2.3. FEC below the transport

In this case, the coding scheme may hide congestion losses from the congestion controller. There are cases where this can drastically reduce the goodput of non-coded flows. Depending on the congestion control, it may be possible to signal to the congestion control mechanism that there was congestion (loss) even when a packet has been recovered, e.g. using ECN, to reduce the impact on the non-coded flows (see Section 6.3.3 and [TENTET]).

6.3. Congestion control and recovered symbols

The objective of this subsection is to describe potential interactions between the congestion control and the recovered symbols.

6.3.1. FEC above the transport

The congestion control may not be able to differentiate repair symbols from actual source packets. The relevance of adding coding at the application layer is related to the needs of the application. For real-time applications, this approach may reduce the number of retransmission. The usage of a non-reliable transport is more adequate in this case.

6.3.2. FEC within the transport

If the two FEC and CC channels are decoupled, the endpoint may exploit different protocols for each channel. The channels may be coupled and one single protocol may be exploited. In both cases, the receiver can differentiate source packets and repair symbols. The receiver may indicate both the number of source packets received and repair symbols that were actually useful in the recovery process of packets.

6.3.3. FEC below the transport

The congestion control may not know what is going on in the network underneath and whether a coding scheme is introduced or not. The congestion control may behave as if no coding scheme is introduced. The only way for a coding channel to indicate that symbols have been recovered is to exploit existing signaling that is understood by the congestion control mechanism. An example would be to indicate to a TCP sender that a packet has been recovered (i.e., congestion has occurred), by using ECN signaling [TENTET].

6.4. Interactions between congestion control and coding rates

This section discusses to what extent the interaction between the congestion control and the coding rates is possible.

6.4.1. FEC above the transport

The coding rate applied at the application layer mainly depends on the available capacity given by the congestion control underneath. Adapting the coding rate to the minimum required data rate of the application may reduce packet losses and improve the quality of experience.

6.4.2. FEC within the transport

In this case, there is an important flexibility in the trade-off, inherent to the use of coding, between (1) reducing goodput when useless repair symbols are transmitted and (2) helping to recover sooner from transmission and congestion losses. As explained in Section 6.3.2, the receiver may indicate to the sender the number of packets that have been received or recovered. The sender may exploit this information to tune the coding ratio. As one example of flexibility of this case, coupling an increased transmission rate with an increasing or decreasing coding rate could be envisioned. A server may use an increasing coding rate as a probe of the channel capacity and adapt the congestion control transmission rate.

6.4.3. FEC below the transport

In this case, the coding rate can be tuned depending on the number of recovered symbols and the rate at which the sender transmits data. The coding scheme is not aware of the congestion control implementation, making it hard for the coding scheme to apply the relevant coding rate.

6.5. On the useless repair symbols

There are cases where useless repair symbols may be transmitted. These impact on the network load and may reduce the goodput of the flow without concrete gains.

6.5.1. FEC above the transport

In this case, the discussion depends on application needs. The only case where adding useless repair symbols does not result in reduced goodput is when the application needs a limited amount of goodput (e.g., VoIP traffic). In this case, the useless repair symbols would only impact the amount of data generated in the network.

6.5.2. FEC within the transport

The sender may exploit the information given by the receiver to reduce the number of useless repair symbols and the resulting goodput reduction.

6.5.3. FEC below the transport

In this case, the useless repair symbols only impact the load of the network without actual gain for the coded flow.

7. Open research questions

This section provides a simplified state-of-the art of the activities related to congestion control and coding. The objective is to identify open research questions and contribute to advice when evaluating coding mechanisms.

7.1. Activities related to congestion control and coding

We map activities related to congestion control and coding with the organization presented in this document:

- o For the FEC above transport case: TBD

- o For the FEC within transport case:
[I-D.swett-nwcr-g-coding-for-quic], [QUIC-FEC], [RFC5109].
- o For the FEC below transport case: [NCTCP],
[I-D.detchart-nwcr-g-tetrys].

7.2. Open research questions

The research questions should be mapped following the organization of this document. In all these three use-cases, open questions remain. There is a general trade-off, inherent to the use of coding, between (1) reducing goodput when useless repair symbols are transmitted and (2) helping to recover from transmission and congestion losses.

For the FEC above transport case, there is a trade-off related to the amount of redundancy to add, as a function of the transport layer protocol and application requirements.

For the FEC within transport case, recovering lost symbols may hide congestion losses to the congestion control. Some existing solutions already propose to disambiguate acked packets from rebuilt packets [QUIC-FEC]. New signalling methods and FEC-recovery-aware congestion controls could be proposed.

For the FEC below transport case, there are opportunities for introducing interaction between congestion control and coding schemes to improve the quality of experience while guaranteeing fairness with other flows. An open question also resides in the relevance of FEC when there are multiple streams that exploit the FEC channel.

7.3. Advices for evaluating coding mechanisms

The contribution to research questions should be mapped following the organization of this document. Otherwise, this may lead to wrong assumptions on the validity of the proposal and wrong ideas about the relevance of coding for a given use case.

The discussion provided in this document aims at encouraging the research community to also consider congestion control aspects when proposing and comparing FEC coding solutions in communication systems. As one example, this draft proposes discussions on the impact of the proposed FEC solution on congestion control, especially loss-based congestion control mechanisms. When a research work aims at improving the throughput by hiding the packet loss signal from the congestion control, the authors should 1) discuss the advantages of using the proposed FEC solution compared to replacing the congestion control by one that ignores a portion of the encountered losses, 2)

critically discuss the impact of hiding packet loss from the congestion control mechanism.

8. Acknowledgements

Many thanks to Spencer Dawkins, Dave Oran, Carsten Bormann, Vincent Roca and Marie-Jose Montpetit for their useful comments that helped improve the document.

9. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

FEC and CC schemes can contribute to DoS attacks. This is not specific to this document.

In case of FEC below the transport, the aggregate rate of source and repair packets may exceed the rate at which a congestion control mechanism allows an application to send. This could result in an application obtaining more than its fair share of the network capacity.

11. Informative References

- [CTCP] Kim (et al.), M., "Network Coded TCP (CTCP)", arXiv 1212.2291v3, 2013.
- [I-D.briscoe-tsvarea-fair]
Briscoe, B., "Flow Rate Fairness: Dismantling a Religion", draft-briscoe-tsvarea-fair-02 (work in progress), July 2007.
- [I-D.detchart-nwcr-g-tetrys]
Detchart, J., Lochin, E., Lacan, J., and V. Roca, "Tetrys, an On-the-Fly Network Coding protocol", draft-detchart-nwcr-g-tetrys-05 (work in progress), February 2020.
- [I-D.swett-nwcr-g-coding-for-quic]
Swett, I., Montpetit, M., Roca, V., and F. Michel, "Coding for QUIC", draft-swett-nwcr-g-coding-for-quic-04 (work in progress), March 2020.
- [NCTCP] Sundararajan (et al.), J., "Network Coding Meets TCP: Theory and Implementation", IEEE INFOCOM 10.1109/JPROC.2010.2093850, 2009.

- [QUIC-FEC] Michel (et al.), F., "QUIC-FEC: Bringing the benefits of Forward Erasure Correction to QUIC", IFIP Networking 10.23919/IFIPNetworking.2019.8816838, 2019.
- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, DOI 10.17487/RFC3758, May 2004, <<https://www.rfc-editor.org/info/rfc3758>>.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <<https://www.rfc-editor.org/info/rfc4340>>.
- [RFC5109] Li, A., Ed., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, DOI 10.17487/RFC5109, December 2007, <<https://www.rfc-editor.org/info/rfc5109>>.
- [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>>.
- [RFC6356] Raiciu, C., Handley, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, DOI 10.17487/RFC6356, October 2011, <<https://www.rfc-editor.org/info/rfc6356>>.
- [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>>.
- [RFC8699] Islam, S., Welzl, M., and S. Gjessing, "Coupled Congestion Control for RTP Media", RFC 8699, DOI 10.17487/RFC8699, January 2020, <<https://www.rfc-editor.org/info/rfc8699>>.
- [TENTET] Lochin, E., "On the joint use of TCP and Network Coding", NWCRG session IETF 100, 2017.

Authors' Addresses

Nicolas Kuhn
CNES

Email: nicolas.kuhn@cnes.fr

Emmanuel Lochin
ENAC

Email: emmanuel.lochin@enac.fr

Francois Michel
UCLouvain

Email: francois.michel@uclouvain.be

Michael Welzl
University of Oslo

Email: michawe@ifi.uio.no

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-nwcrgr-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 Mr, 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 (TO octets). A DDP packet has totally $8+TO$ octets.

2.4.1. Packet Header

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

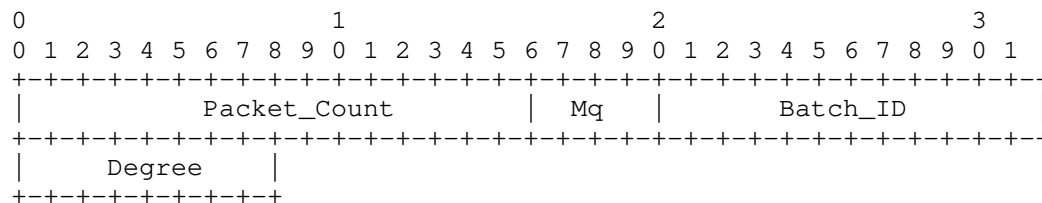


Figure 3: BATS packet header format.

- o Packet_Count: 16-bit unsigned integer, specifying the number K of packets of the BATS session.
- o Mq: 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

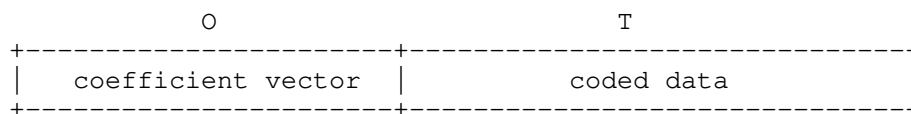


Figure 4: BATS packet payload format.

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

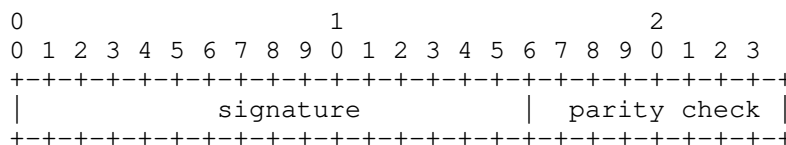


Figure 5: BATS packet footer format.

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