Workgroup: Internet Engineering Task Force
Internet-Draft:
draft-kuhn-quic-bdpframe-extension-02
Published: 10 May 2023
Intended Status: Standards Track
Expires: 11 November 2023
Authors: N. Kuhn
Thales Alenia Space
E. Stephan
Orange
G. Fairhurst
University of Aberdeen
C. Huitema
Private Octopus Inc.

BDP Frame Extension

Abstract

This document describes the BDP_FRAME extension for QUIC. The frame enables the exchange of Congestion Control (CC) parameters related to the path characteristics between the receiver and the sender during a connection. These CC parameters can be utilised by the Careful Resume method when a new connection is established or for application-limited traffic. The CC parameters allows a receiver to prepare to use the additional capacity that could be made available when the method is used. This CC parameters can also be used by the receiver to request that previously computed CC parameters related to the path characteristics, are not used, when the receiver has additional information about the path or traffic to be sent.

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 11 November 2023.

Table of Contents

1. Introduction

This document extends the Careful Resume method [I-D.kuhn-tsvwg-careful-resume] by allowing sender-generated CC parameters to be stored at the receiver. By transfering the CC parameters to a receiver, it also releases the sender from needing to retain CC parameter state for each receiver. This specifically allows a receiver to implement a policy that informs a sender whether the receiver desires the sender to reuse the CC parameters.

This document defines the method to exchange the CC parameters between a QUIC receiver and the sender in an interoperable manner. The process is outlined here:

1. For an established connection, the current RTT (current_rtt), bottleneck bandwidth (current_bb) and current receiver Endpoint Token (current_endpoint_token) are stored as saved_rtt, saved_bb and saved_endpoint_token within a BDP_FRAME. The sender computes a secured hash with its own selection of the CC parameters of the BDP_FRAME, encrypts the hash and sends this within the BDP_FRAME. The sender encrypts the hash so that the receiver can not read nor modify the content of the secured hash ;

2. The receiver can read the non-encrypted portion of the BDP_FRAME parameters, but is not permitted to modify any CC parameters. The receiver is unable to read the hash.

3. A receiver later sends a BDP-FRAME back to the sender to re-use previously computed CC parameters;

4. The sender is then able to utilise the CC parameters in the BDP_FRAME in new connection to the same endpoint.

This method can apply to any resumed QUIC session: both a saved_session and a recon_session can be a 0-RTT QUIC connection or a 1-RTT QUIC connection.

1.1. Optimizing Client Requests

Where the receiver is aware of a high Bandwidth-Delay Product (BDP), it can adapt other CC parameters to better utilize the available capacity, such as increasing the value of flow control pararemeters.

Some designs of application do not use long-lasting transport connections. Instead, they use a series of shorter connections, typically each using the same path. This style of application can benefit from this method, and could be enhanced by allowing the application to receive an estimate of the expected characteristics, which could help to appropriately use the new connection (e.g., adapting the content of quality for a video application; or anticipating the time taken to complete the transmission of an object).

This paragraph considers a scenario where a client uses Dynamic Adaptive Streaming over HTTPS (DASH). Such a client might be unable to receive sufficient data to reach the video playback quality that is supported by the path, because for each video chunk, the transport protocol needs to independently determine the path capacity. The lower transfer rate is safe, but can also lead to an

overly conservative requested rate to the sender, because clients often adapt their application-layer requests based on the transport performance (i.e., the client could fail to increase the requested quality of video chunks, or to fill buffers to avoid stalling playback or to send high quality advertisements).

When using Dynamic Adaptive Streaming over HTTPS (DASH), clients might encounter issues in knowing the available path capacity or DASH can encounter issues in reaching the best available video playback quality. The client requests could then be adapted and specific traffic could utilize the previously observed path characteristics (such as encouraging the client to increase the quality of video chunks, to fill the buffers and avoid video blocking or to send high quality adds).

1.2. Three approaches

This section reviews three approaches to implement Careful Resume.

(1) The saved CC parameters are stored at the sender ("Local storage") and is never sent to a receiver;

(2) Some CC parameters are transmitted to the receiver, which can be used when reconnecting, but the receiver cannot read the CC parameters received from the sender ("NEW TOKEN");

(3) the saved CC parameters are transmitted to a receiver, which can use it when reconnecting. The receiver can read the CC parameters to accept or not the use of CC parameters (a.k.a. "BDP extension").

1.2.1. Independent Local Storage of Values

This approach independently lets both a receiver and a sender store their CC parameters:

* During a 1-RTT session, the endpoint stores the RTT (as the saved_rtt) and bottleneck bandwidth (as the saved_bb) together in the session resume ticket.

* The sender maintains a table of previously issued tickets, indexed by the random ticket identifier that is used to guarantee uniqueness of the Authenticated Encryption with Associated Data (AEAD) encryption. Old tokens are removed from the table using the Least Recently Used (LRU) logic. For each ticket identifier, the table holds the RTT and bottleneck bandwidth (i.e. saved_rtt and saved_bb), and also the Endpoint Token of the receiver (i.e. saved_endpoint_token).

*During the new session establishment (0-RTT or 1-RTT), the local endpoint waits for the first RTT measurement from the remote peer. This is used to verify that the current_rtt has not significantly changed from the saved_rtt (used as an indication that the CC parameters are appropriate for the current path).

*If this RTT is confirmed, the endpoint also verifies that an IW of data has been acknowledged without requiring retransmission or resulting in an ECN CE-mark. This second check detects whether a path is experiencing significant congestion (i.e., where it would not be safe to update the cwnd based on the saved_bb). In practice, this could be realized by a proportional increase in the cwnd, where the increase is (saved_bb/ IW)*proportion_of_IW_currently-ACKed.

This solution does not allow a receiver to request the sender not to use the CC parameters in the BDP Frame. If the sender does not want to store the metrics from previous connections, an equivalent of the tcp_no_metrics_save for QUIC may be necessary. This option could be negotiated that allows a receiver to choose whether to use the saved CC parameters.

1.2.2. Using NEW_TOKEN frames

A sender can send a NEW_TOKEN Frame to the receiver. The token is an opaque (encrypted) blob and the receiver can not read its content (see section 19.7 of [RFC9000]). The receiver sends the received token in the header of an Initial packet of a later connection.

1.2.3. BDP Frame

Using BDP Frames, the sender could send a set of CC parameters to the receiver. The use of the BDP Frame is negotiated with the receiver. The receiver can read its content. If the receiver permits using the previous CC parameters, it can send the BDP Frame back to the sender in an Initial packet of a later connection.

2. Notations and Terms

*BDP: defined below

*CWND: the congestion window used by a sender (maximum number of bytes allowed in flight by the CC)

*current_bb : Current estimated bottleneck bandwidth

*saved_bb: Estimated bottleneck bandwidth preserved from a previous connection

*RTT: Round-Trip Time

*current_rtt: Current RTT

*saved_rtt: RTT preserved from a previous connection

*endpoint_token : Endpoint Token of the receiver

*current_endpoint_token : Current Endpoint Token of the receiver

*saved_endpoint_token : Endpoint Token of the receiver preserved from a previous connection

*remembered CC parameters: a combination of saved_rtt and saved_bb

*secured hash : hash generated by the sender using a list of CC parameters that it selected. The sender uses a private key to protect the hash.

[RFC6349] defines the BDP as follows: "Derived from Round-Trip Time (RTT) and network Bottleneck Bandwidth (BB), the Bandwidth-Delay Product (BDP) determines the Send and Received Socket buffer sizes required to achieve the maximum TCP Throughput." This document considers the BDP estimated by a sender for the path to the receiver. This includes all buffering along this network path. The estimated BDP is related to the volume of bytes in flight and the measured path RTT.

A QUIC connection is allowed to use the procedure detailed in [RFC6349] to measure the BDP, but is permitted to choose another method [RFC9002].

A sender might be able to also utilise other information to estimate the BDP. Congestion controllers, such as CUBIC or RENO, could estimate the saved_bb and current_bb values by combining the cwnd/ flight_size and the minimum RTT. A different method could be used to estimate the same values when using a rate-based congestion controller, such as BBR [I-D.cardwell-iccrg-bbr-congestion-control].

It is important to consider whether a method could result in over-estimating the bottleneck bandwidth, and the preserved values therefore ought to be used with caution.

2.1. Requirements Language

The keywords "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.

Variable-length integer encoding is defined in section 16 of [RFC9000].

3. BDP Frame

This section describes the use of a new Frame, the BDP_FRAME. The BDP_FRAME can be utilized by the congestion controller and its data is not be limited by flow control limits. The sender and the receiver MAY send multiple BDP_FRAMEs in both 1-RTT and 0-RTT connections. The rate of update SHOULD be limited (e.g. much less frequent than once for several RTTs).

3.1. BDP_FRAME Format

A BDP_FRAME is formatted as shown in Figure 1.

```
BDP_FRAME {
  Type (i) = 0xXXX,
  Hash (...),
  Lifetime (i),
  Saved BB (i),
  Saved RTT (i),
  Saved Endpoint Token (...)
}
```

Figure 1: BDP_FRAME Format

A BDP_FRAME contains the following fields:

*Hash (secured_hash): The secured_hash is generated by the sender using CC parameters from the BDP_FRAME. The sender encrypts the hash so that the receiver can not read it.

*Lifetime (extension_lifetime): The extension_lifetime is a value in milliseconds, encoded as a variable-length integer. This follows the design of a NewSessionTicket of TLS [RFC8446]. This represents the validity in time of this extension.

*Saved BB (saved_bb): The saved_bb is a value in bytes, encoded as a variable-length integer. The bottleneck bandwidth can be estimated for the previous connection by the sender. Using the previous values of bytes_in_flight defined in [RFC9002] can result in overshoot of the bottleneck capacity, and ought to be used carefully. It is advised to not only use the amount of bytes in flight but also the goodput.

*Saved RTT (saved_rtt): The saved_rtt is a value in milliseconds, encoded as a variable-length integer. The saved_rtt can be set to the min_rtt. NOTE: The min_rtt defined in [RFC9002], does not

track a decreasing RTT: therefore the min_rtt reported might be larger than the actual minimum RTT measured during the 1-RTT connection.

*Saved Endpoint Token (saved_endpoint_token) : The saved_endpoint_token (More details in [I-D.kuhn-tsvwg-careful-resume]).

Note: The Endpoint Token is defined in [I-D.kuhn-tsvwg-careful-resume], and is discussed in the context of this protocol exchange in a later section.

3.1.1. Extension activation

The receiver can accept the transmission of BDP_FRAMEs from the sender by using the enable_bdp transport extension.

enable_bdp (0xTBD): in the 1-RTT connection, the receiver indicates to the sender that it wishes to receive BDP extension Frames. The default value is 0. In this specification, enable_bdp values larger than 3 are reserved for future, and the receipt of these values MUST be treated as a connection error of type TRANSPORT_PARAMETER_ERROR [RFC9000].

*0: Default value. If the receiver does not send an enable_bdp parameter, the sender considers that the receiver does not support, or does not wish to activate, the BDP extension.

*1: The receiver indicates to the sender that it wishes to receive BDP_FRAMEs and activates the optimization.

*2: The receiver indicates that it does not wish to receive BDP Frames but activates the optimization.

*3: The receiver indicates that it wishes to receive BDP_FRAMEs, but does not activate the optimization.

This Transport Parameter is encoded as described in Section 18 of [RFC9000].

If the receiver activates the extension, it agrees to receive and read BDP_FRAMEs. If the receiver activates the optimization, it allows the sender to utilise the previously computed CC parameters. The receiver could then agree to do session resumption optimization without actually reading the previous CC parameters.

3.1.2. Using the CC parameters with Care

Care is needed in the use of any temporal information to assure safe use of the Internet and to be robust to changes in traffic patterns,

network routing and link/node failures. There are also cases where using the CC parameters of a previous connection are not appropriate, and a need to evaluate the potential for malicious use of the method. The specification for the QUIC transport protocol [RFC9000] therefore notes "Generally, implementations are advised to be cautious when using previous values on a new path."

Careful exploitation of the CC parameters is discussed in [I-D.kuhn-tsvwg-careful-resume].

3.2. Discussion

A receiver using the BDP_FRAME extension has the choice of accepting the reuse of the previous CC parameters, or requesting the sender to not reuse the previous CC parameters.

This extension MUST NOT provide an opportunity for the current connection to be a vector for an amplification attack. The QUIC address validation process, used to prevent amplification attacks, SHOULD be performed [RFC9000].

The CC parameters are measured by the sender during a previous connection to the same receiver. The BDP extension is protected by the mechanism that protects the exchange of the 0-RTT transport parameters. For version 1 of QUIC, the BDP extension is protected using the mechanism that already protects the "initial_max_data" parameter. This is defined in sections 4.5 to 4.7 of [RFC9001]. This provides a way for the sender to verify that the CC parameters proposed by the receiver are the same as those that the sender sent to the receiver during a previous connection.

The sender SHOULD NOT trust the content of the BDP Frame received from the receiver. Even if the QUIC packets containing the BDP Frame are encrypted, a receiver could modify the values within the extension and encrypt the QUIC packet. One way to avoid this is for a sender to also store the saved_rtt and saved_bb parameters. Another way to avoid this is to use the secured hash generated by the sender. If the receiver modifies a CC parameter, the result of the hash would be different. The sender should then avoid exploiting previously estimated CC parameters.

An example of implementation where the sender computes an Endpoint Token that seeks to uniquely identify the receiver is provided in Section 4.1. Implementation details are being left independent from the specification of BDP-FRAME.

3.3. Interoperability and Use Cases

A sender that stores a resumption ticket for each receiver to protect against replay on a third party, could also store the

Endpoint Token (i.e., saved_endpoint_token) and CC parameters (i.e., saved_rtt and saved_bb) of a previous connection.

When the BDP Frame extension is used, locally stored CC parameters at the sender can provide a cross-check of the CC parameters sent by a receiver. The sender can anyway enable a safe jump, but without the BDP Frame extension. However, using the CC parameters enables a receiver to choose whether to request this or not, enabling it to utilize local knowledge of the network conditions, connectivity, or connection requirements.

Four cases are identified:

1. The network path has changed and the new path is different. This might be evident from a change of local interface, a change in the client or sender IP address, or similar indication from the network. Using the saved CC parameters could increase congestion.

2. The network path has changed, but the new path is not known to be different. This case might be accompanied by a change in the RTT, or evident by loss observed at the start of the new connection and the saved CC parameters is not appropriate.

3. The network conditions have changed and it is discovered that the current capacity is less than the previously estimated bottleneck bandwidth. Using the saved CC parameters would then increase congestion, and the flow needs to adjust to a lower safe rate;

4. The stored CC parameters is too old. In this case, it is no longer be reasonable to expect the path to have same characteristics, and the the saved CC parameters is no longer appropriate.

In all these case, the Careful Resume method is not be used, and a sender needs to return to a normal CC behavior. The method can still be used to characterize the new path, enabling later flows to use this method.

{XXX-Editor-note: Text to be improved: Storing local values related to the BDP would help improve the ingress for new connections, however, not using a BDP Frame extension could reduce the interest of the approach where (1) the receiver knows the BDP estimation at the sender, (2) the receiver decides to accept or reject ingress optimization, (3) the receiver tunes application level requests.}

4. Identifying the Path

In a simple network scenario, the sending endpoint could use the IP source address to identify a path. This could work when one globally-allocated IP address is set per interface. There are many cases where the IP address would not an acceptable to identify a path. Section 8 of [RFC9040] describes cases where the IP address is not a suitable value when performing TCP control block sharing. In general the IP address of the sender is made public in the network-layer header of IP packets. When sharing internal state, [RFC6973] identifies relevant privacy considerations.

Examples of network uses where a source address is not a suitable endpoint token include:

* The sending endpoint might not be identifiable remotely from its IP address because a device on the network path translates the address using a form of NAT/NAPT. In this case, a private IP address might be used, which does not identify a specific endpoint.

* In some cases, a sender can choose to vary the source address over time to avoid linkability in the observable IP header, e.g., because the used source address embeds private information, such as the endpoint's MAC address/EID.

Note: There are use-cases where for the purpose of identifying a path, the token does not need to be globally unique, but needs to be sufficiently unique to prevent attempts to misrepresent the path being used such as an attack on the congestion controller. Using a smaller size of token can add to the ambiguity set, reducing this likability.

NOTE: A different Endpoint Token is used for each direction of transmission. A receiver might decide not to provide an Endpoint Token to a sender, to avoid exposing additional linkable information (but also preventing use of any mechanism that relies on the token).

4.1. Example use of an Endpoint Token

The sender computes an Endpoint Token that seeks to uniquely identify the path that it uses to communicate with the receiver (1) this is associated with the path information it sends. The Endpoint Token ought to be encrypted to avoid sending linkable information observable to eavesdroppers on the path. The receiver stores the path information together with the Endpoint Token, together with the sender's address/name (2). When the receiver later wishes the sender to use the stored path information it returns the information to the sender (3) together with the Endpoint Token. The sender recomputes

the Endpoint Token and compares this with the received Endpoint Token before using the CC parameters.

1. The Sender transmits the Endpoint Token to the Receiver

2. The Receiver holds an Endpoint Token

3. The Receiver transmits the Endpoint Token to the Sender

4.2. Security Related to use of the Endpoint Token

A number of security-related topics have been discussed, mostly concerning the potential exposure of the identity on the path. This information can also be visible in the IP source address or higher-layer data, but can be hidden from a remote endpoint using methods such as MASQUE proxy. When used to inform the transport system using a layered proxy, the transport endpoint token refers to the endpoints of the outer QUIC header, and hence the proxy itself, not the end-to-end communication relayed by the proxy.

A sender might decide to not use this method if it has a strong requirement to prevent flows being linkable with previous flows to the same endpoint. A decision not to provide an Endpoint Token necessarily prevents the sender from requesting the receiver to return path information to allow the same CC parameters to be re-used, potentially strengthening privacy but consequently eliminating any performance benefits.

5. Acknowledgments

The authors would like to thank Gabriel Montenegro, Patrick McManus, Ian Swett, Igor Lubashev, Robin Marx, Roland Bless and Franklin Simo for their fruitful comments on earlier versions of this document.

The authors would like to particularly thank Tom Jones for co-authoring previous versions of this document.

6. IANA Considerations

{XXX-Editor note: Text is required to register the BDP Frame and the enable_bdp transport parameter. Parameters are registered using the procedure defined in [RFC9000].}

TBD: Text is required to register the BDP_FRAME and the enable_bdp transport parameter. Parameters are registered using the procedure defined in [RFC9000].

7. Security Considerations

Security considerations for the CC method are discussed in the Security Considerations section of Careful Resume.

7.1. Protecton from Malicious Receivers

The sender MUST check the integrity of the saved_rtt and saved_bb parameters received from a receiver.

There are several solutions to avoid attacks by malicious receivers:

*Solution #1 : The sender stores a local estimate of the bottleneck bandwidth and RTT parameters as the saved_bb and saved_rtt.

*Solution #2 : The sender sends the estimate of the bottleneck bandwidth and RTT parameters to the receiver as the saved_bb and saved_rtt in a block of CC parameters that is authenticated. These CC parameters also could be encrypted by the sender. The receiver resends the same CC parameters for a new connection. The sender can use its local key information to authenticate the CC parameters, without needing to keep a local copy.

*Solution #3 : This approach is the same as above, except that the sender provides an estimate of the saved_rtt and saved_bb parameters in a form that may be read by the receiver. Using the security mechanisms provided in this document, the sender can verify that the receiver did not change the CC parameters inside the frame. The receiver can read, but not modify, the saved_rtt and saved_bb parameters and could enable a receiver to decide whether the new CC parameters are thought appropriate, based on receiver-side information about the network conditions, connectivity, or needs of the new connection.

7.2. Rationale behind the different implementation options

The NewSessionTickets message of TLS can offer a solution. The proposal is to add a 'bdp_metada' field in the NewSessionTickets, which the receiver is able to read. The only extension currently defined in TLS1.3 that can be seen by the receiver is max_early_data_size (see Section 4.6.1 of [RFC8446]). However, in the general design of QUIC, TLS sessions are managed by a TLS stack.

Three distinct approaches are presented: sending an opaque blob to the receiver that the receiver may return to the sender when establishing a future new connection (see Section 1.2.2), enabling local storage of the CC parameters (see Section 1.2.1) and a BDP Frame extension (see Section 1.2.3).

8. References

8.1. 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>.

[RFC6349] Constantine, B., Forget, G., Geib, R., and R. Schrage, "Framework for TCP Throughput Testing", RFC 6349, DOI 10.17487/RFC6349, August 2011, <https://www.rfc-editor.org/info/rfc6349>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/info/rfc6973>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/info/rfc9001>.

[RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <https://www.rfc-editor.org/info/rfc9002>.

[RFC9040] Touch, J., Welzl, M., and S. Islam, "TCP Control Block Interdependence", RFC 9040, DOI 10.17487/RFC9040, July 2021, <https://www.rfc-editor.org/info/rfc9040>.

8.2. Informative References

[I-D.cardwell-iccrg-bbr-congestion-control] Cardwell, N., Cheng, Y., Yeganeh, S. H., Swett, I., and V. Jacobson, "BBR Congestion Control", Work in Progress,

Internet-Draft, draft-cardwell-iccrg-bbr-congestion-control-02, 7 March 2022, <https://datatracker.ietf.org/doc/html/draft-cardwell-iccrg-bbr-congestion-control-02>.

[I-D.kuhn-tsvwg-careful-resume] Kuhn, N., Emile, S., Fairhurst, G., and C. Huitema, "Careful convergence of congestion control from retained state with QUIC", Work in Progress, Internet-Draft, draft-kuhn-tsvwg-careful-resume-00, 3 March 2023, <https://datatracker.ietf.org/doc/html/draft-kuhn-tsvwg-careful-resume-00>.

Appendix A. Comparing BDP-Frame Solutions

Rationale	Solution	Advantage	Drawback	Comment
#2 Malicious receiver	#1 Local storage	Enforced security	A receiver is unable to reject Malicious sender could fill a receive buffer Limited use-cases	Section 4.2
	#2 NEW_TOKEN	Save resource at sender Opaque token protected	A malicious receiver could change token even if protected A malicious sender could fill the receive buffer sender may not trust receiver	Section 4.3
	#3 BDP extension	Extended use-cases Save resource at sender A receiver can read and decide to reject BDP extension protected	A malicious receiver could change BDP even if protected A sender may not trust a receiver	Section 4.4

{XXX-Editor-Note: Need to clarify the text around changing the authenticated token.}

Figure 2: Comparing BDP-Frame Solutions

Authors' Addresses

Nicolas Kuhn

Thales Alenia Space

Email: nicolas.kuhn.ietf@gmail.com

Emile Stephan
Orange

Email: emile.stephan@orange.com

Godred Fairhurst
University of Aberdeen
Department of Engineering
Fraser Noble Building
Aberdeen
AB24 3UE
United Kingdom

Email: gorry@erg.abdn.ac.uk

Christian Huitema
Private Octopus Inc.

Email: huitema@huitema.net