University of Glasgow
School of Computing Science

Parsing Protocol Standards to Parse Standard Protocols

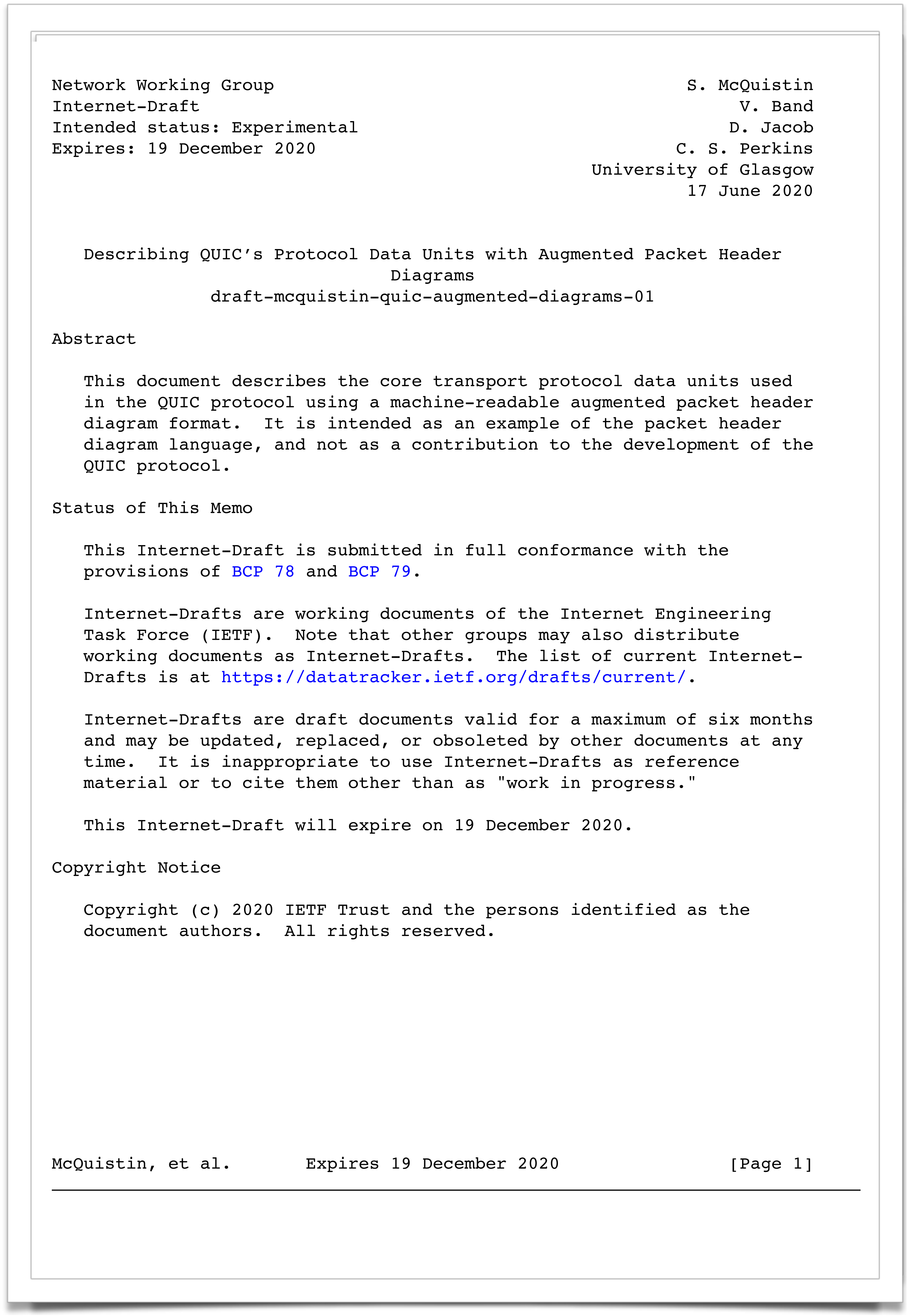
Stephen McQuistin
Vivian Band
Dejice Jacob
Colin Perkins

Applied Networking Research Workshop 2020

This work is funded by the UK Engineering and Physical Sciences Research Council, under grant EP/R04144X/1.

Parsing Protocol Standards …

- Internet standards documents are typically written in English prose
- As protocols become more complex, this becomes undesirable
- Inconsistencies and ambiguities are easily introduced by natural language descriptions
- Formal specification languages would make documents more concise and consistent, and enable machine parsing



Network Working Group — S. McQuistin
Internet-Draft — V. Band
Intended status: Experimental — D. Jacob
Expires: 19 December 2020 — C. S. Perkins
University of Glasgow
17 June 2020

Describing QUIC's Protocol Data Units with Augmented Packet Header Diagrams
draft-mcquistin-quic-augmented-diagrams-01

Abstract

This document describes the core transport protocol data units used in the QUIC protocol using a machine-readable augmented packet header diagram format. It is intended as an example of the packet header diagram language, and not as a contribution to the development of the QUIC protocol.

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 19 December 2020.

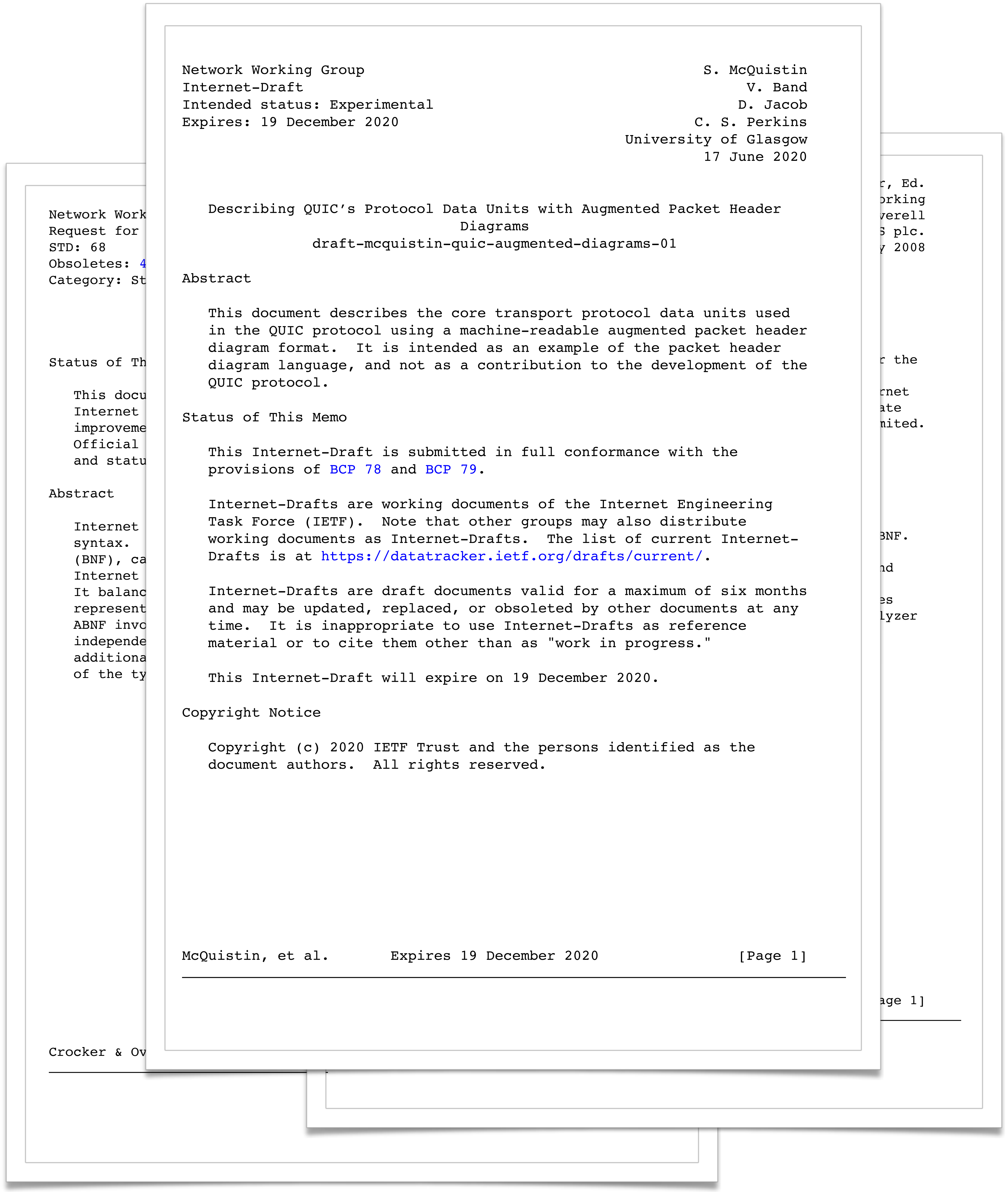
Copyright Notice

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

McQuistin, et al. Expires 19 December 2020 [Page 1]

Parsing Protocol Standards …

- Internet standards documents are typically written in English prose
- As protocols become more complex, this becomes undesirable
- Inconsistencies and ambiguities are easily introduced by natural language descriptions
- Formal specification languages would make documents more concise and consistent, and enable machine parsing



Network Working Group S. McQuistin
Internet-Draft V. Band
Intended status: Experimental D. Jacob
Expires: 19 December 2020 C. S. Perkins
University of Glasgow
17 June 2020

Describing QUIC's Protocol Data Units with Augmented Packet Header Diagrams
draft-mcquistin-quic-augmented-diagrams-01

Abstract

This document describes the core transport protocol data units used in the QUIC protocol using a machine-readable augmented packet header diagram format. It is intended as an example of the packet header diagram language, and not as a contribution to the development of the QUIC protocol.

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 19 December 2020.

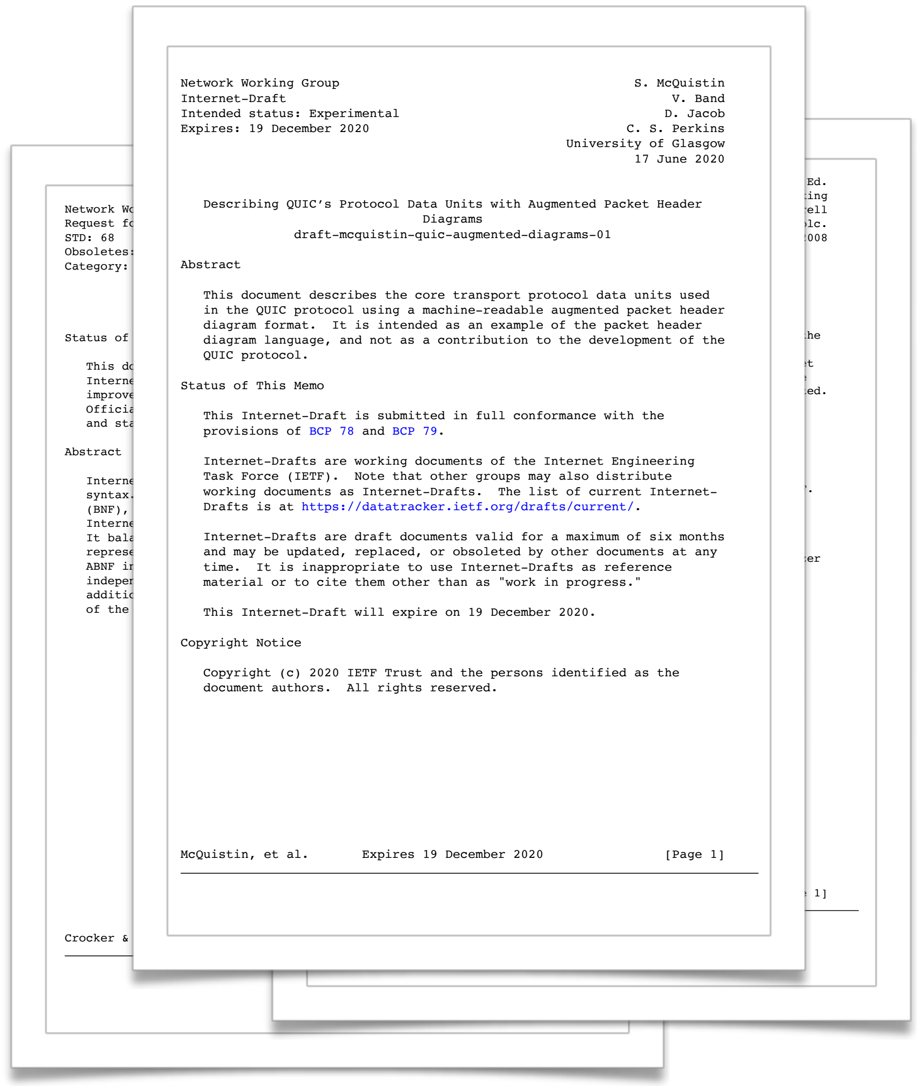
Copyright Notice

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

McQuistin, et al. Expires 19 December 2020 [Page 1]

… to Parse Standard Protocols

- Machine readability would enable automatic code generation
- This enables testing of the protocol specification as it develops
- Modern, secure systems languages can be supported
- Overall, the security and trustworthiness of standards may be improved


Network Working Group S. McQuistin
Internet-Draft V. Band
Intended status: Experimental D. Jacob
Expires: 19 December 2020 C. S. Perkins
University of Glasgow
17 June 2020

Describing QUIC's Protocol Data Units with Augmented Packet Header Diagrams
draft-mcquistin-quic-augmented-diagrams-01

Abstract

This document describes the core transport protocol data units used in the QUIC protocol using a machine-readable augmented packet header diagram format. It is intended as an example of the packet header diagram language, and not as a contribution to the development of the QUIC protocol.

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/.

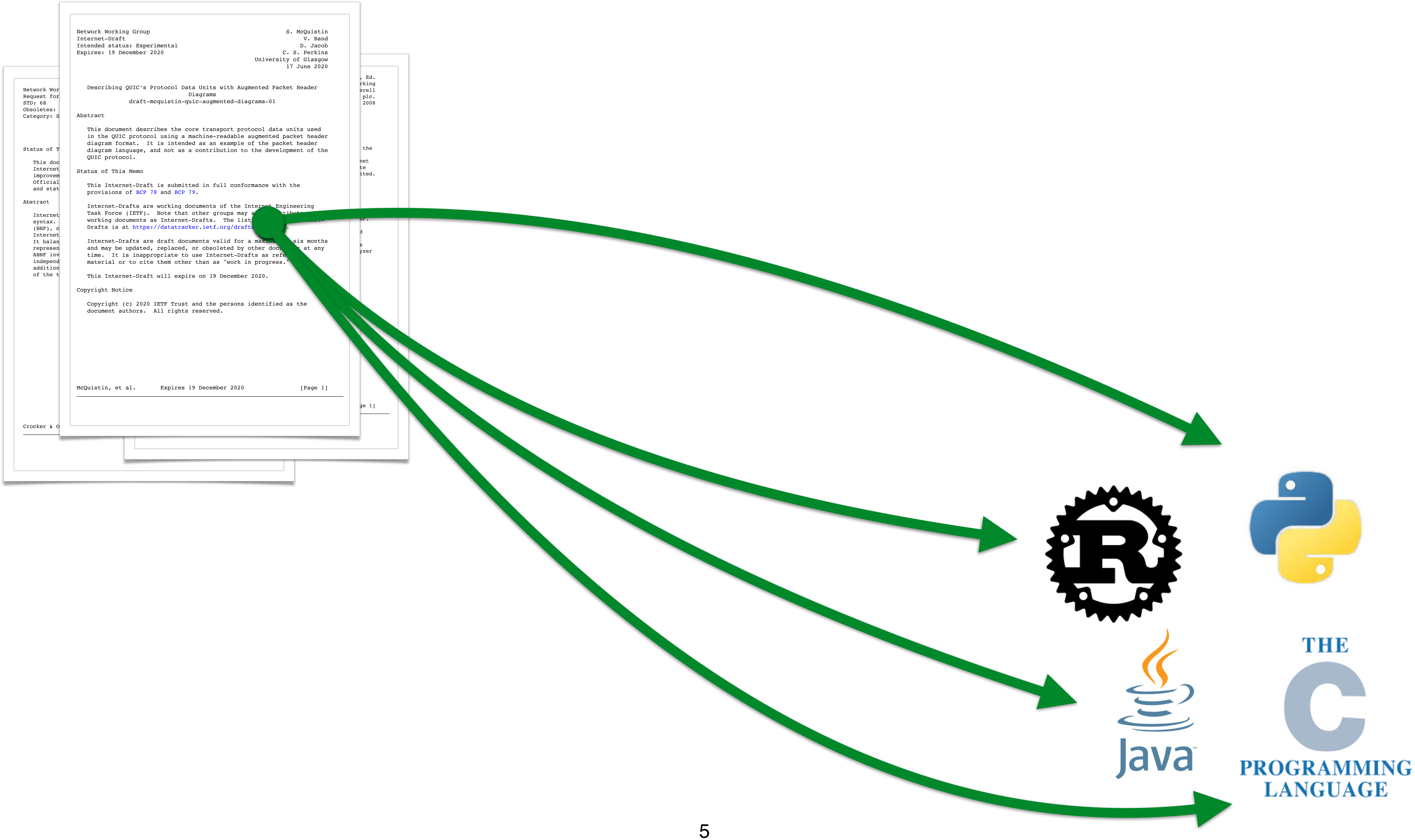
Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

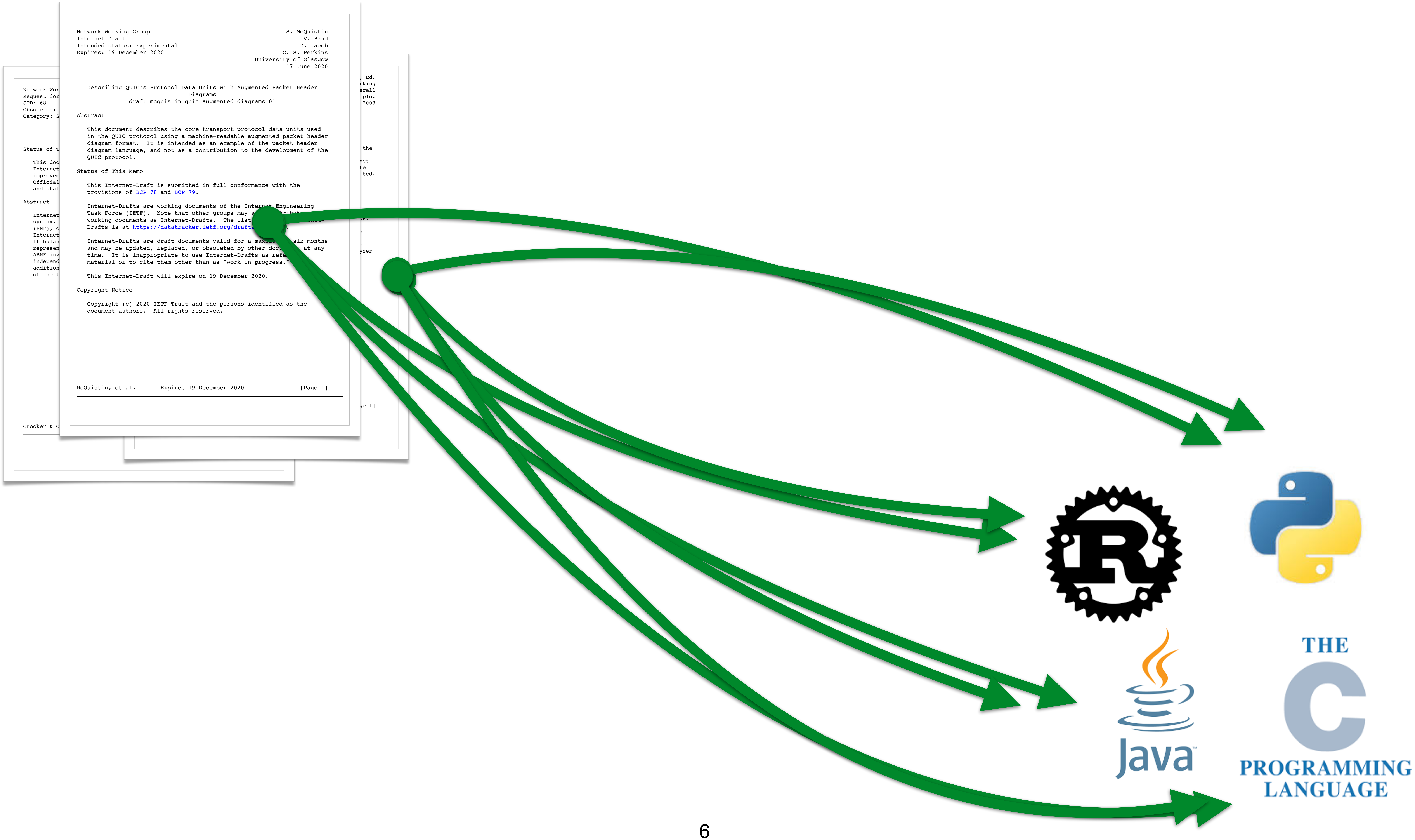
This Internet-Draft will expire on 19 December 2020.

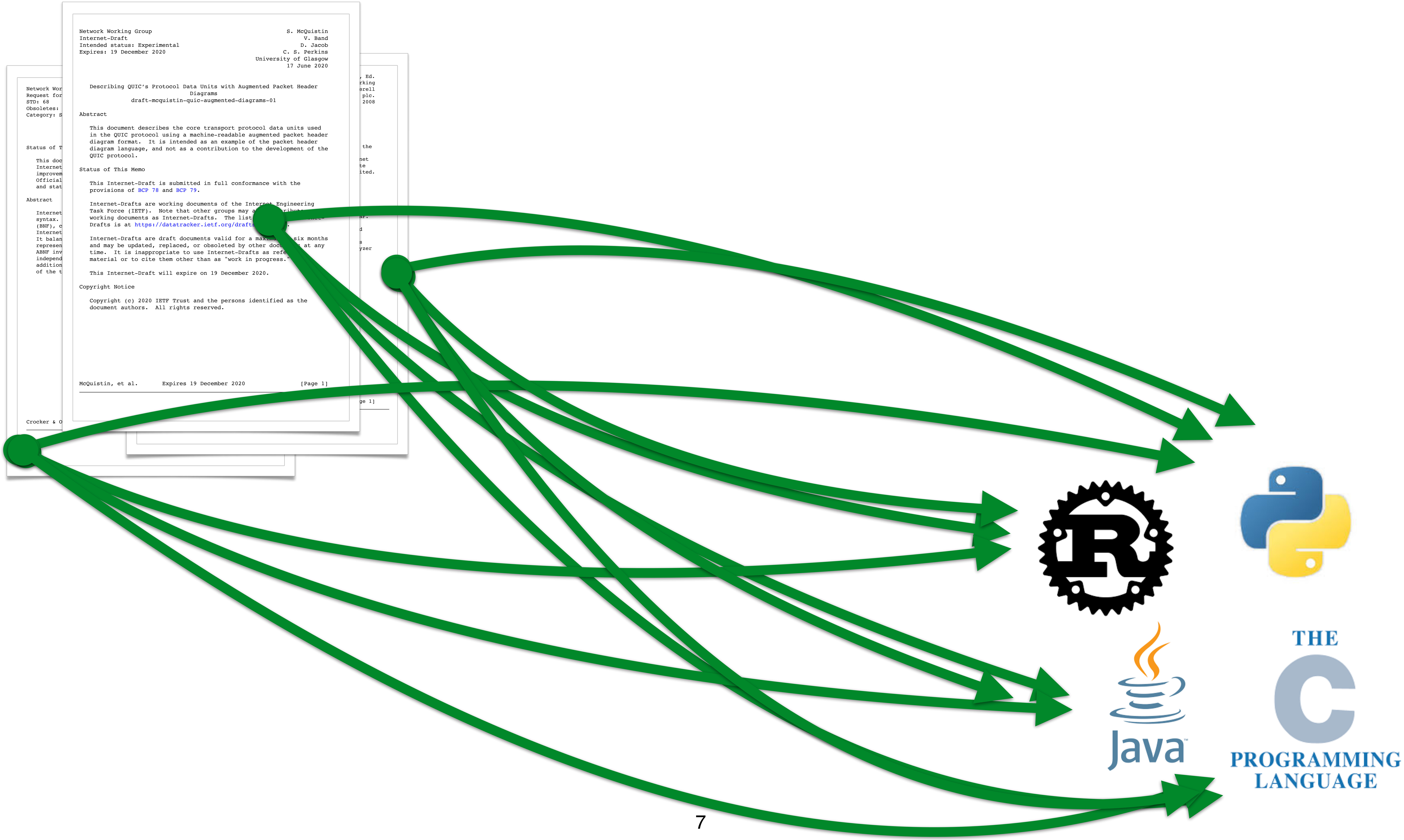
Copyright Notice

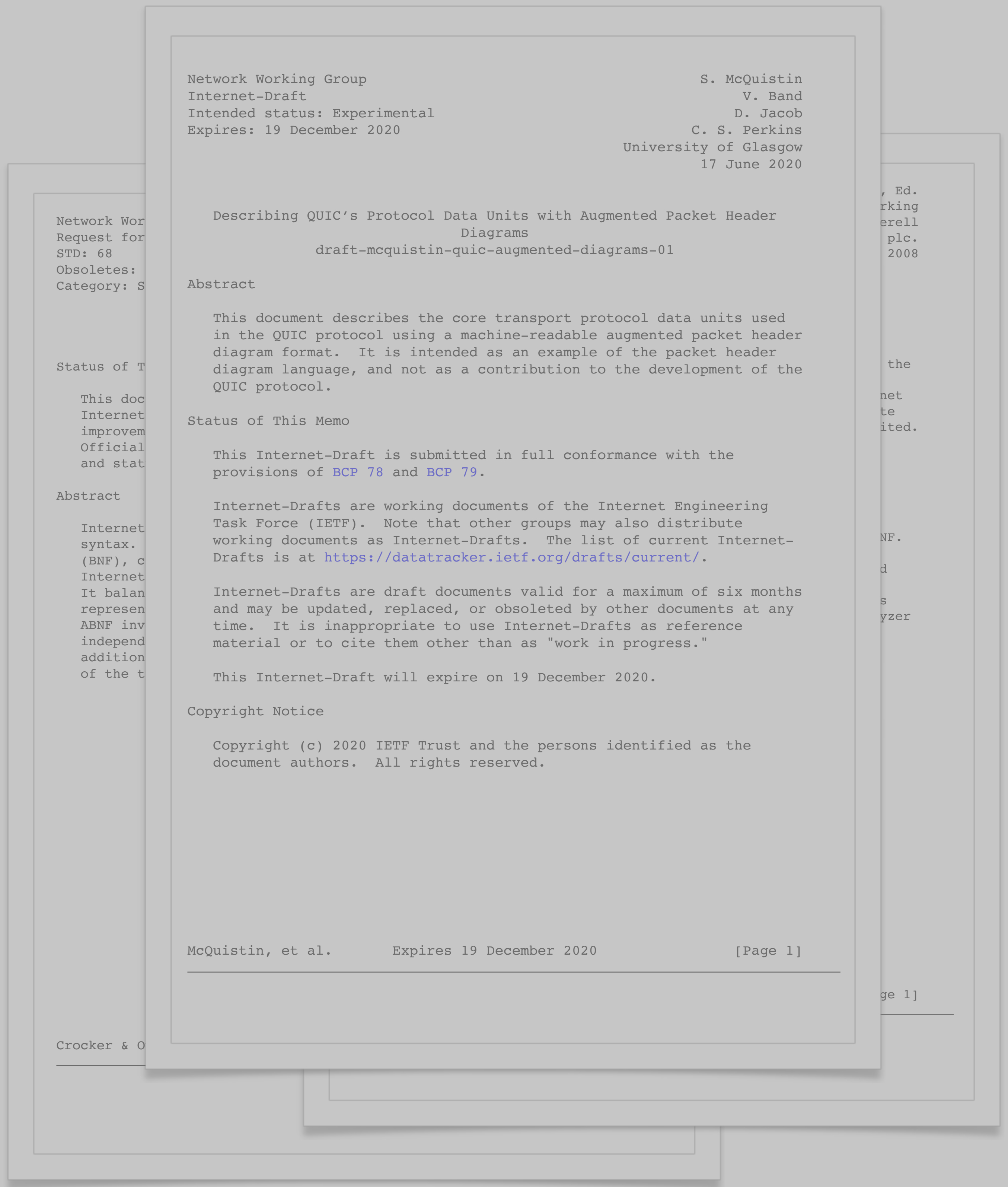
Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

McQuistin, et al. Expires 19 December 2020 [Page 1]


Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 19 December 2020
S. McQuistin
V. Band
D. Jacob
C. S. Perkins
University of Glasgow
17 June 2020
Describing QUIC's Protocol Data Units with Augmented Packet Header Diagrams
draft-mcquistin-quic-augmented-diagrams-01
Abstract
This document describes the core transport protocol data units used in the QUIC protocol using a machine-readable augmented packet header diagram format. It is intended as an example of the packet header diagram language, and not as a contribution to the development of the QUIC protocol.
Status of This Memo
This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.
This Internet-Draft will expire on 19 December 2020.
Copyright Notice
Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.
McQuistin, et al. Expires 19 December 2020 [Page 1]
Java
THE
C
PROGRAMMING
LANGUAGE


Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 19 December 2020
S. McQuistin
V. Band
D. Jacob
C. S. Perkins
University of Glasgow
17 June 2020
Describing QUIC's Protocol Data Units with Augmented Packet Header Diagrams
draft-mcquistin-quic-augmented-diagrams-01
Abstract
This document describes the core transport protocol data units used in the QUIC protocol using a machine-readable augmented packet header diagram format. It is intended as an example of the packet header diagram language, and not as a contribution to the development of the QUIC protocol.
Status of This Memo
This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.
This Internet-Draft will expire on 19 December 2020.
Copyright Notice
Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.
McQuistin, et al. Expires 19 December 2020 [Page 1]
Java
THE
C
PROGRAMMING
LANGUAGE


Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 19 December 2020
S. McQuistin
V. Band
D. Jacob
C. S. Perkins
University of Glasgow
17 June 2020
Describing QUIC's Protocol Data Units with Augmented Packet Header Diagrams
draft-mcquistin-quic-augmented-diagrams-01
Abstract
This document describes the core transport protocol data units used in the QUIC protocol using a machine-readable augmented packet header diagram format. It is intended as an example of the packet header diagram language, and not as a contribution to the development of the QUIC protocol.
Status of This Memo
This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.
This Internet-Draft will expire on 19 December 2020.
Copyright Notice
Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.
McQuistin, et al. Expires 19 December 2020 [Page 1]
Java
THE C PROGRAMMING LANGUAGE



Network Working Group S. McQuistin
Internet-Draft V. Band
Intended status: Experimental D. Jacob
Expires: 19 December 2020 C. S. Perkins
University of Glasgow
17 June 2020

Describing QUIC's Protocol Data Units with Augmented Packet Header Diagrams
draft-mcquistin-quic-augmented-diagrams-01

Abstract

This document describes the core transport protocol data units used in the QUIC protocol using a machine-readable augmented packet header diagram format. It is intended as an example of the packet header diagram language, and not as a contribution to the development of the QUIC protocol.

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 19 December 2020.

Copyright Notice

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

McQuistin, et al. Expires 19 December 2020 [Page 1]

A common protocol representation

What are the requirements of a common protocol representation?

Representing Protocol Data

- **Syntax description languages**
  - ABNF, ASN.1, the TLS 1.3 presentation language, …

Representing Protocol Data

- **Syntax description languages**
  - ABNF, ASN.1, the TLS 1.3 presentation language, …

These languages can only be used to describe protocol syntax

Representing Protocol Data

- **Syntax description languages**
  - ABNF, ASN.1, the TLS 1.3 presentation language, …
- **Protocol type systems**
  - eTPL, YANG, NetPDL, PADS, DataScript, PacketTypes, the Meta Packet Language, …

Representing Protocol Data

- **Syntax description languages**
  - ABNF, ASN.1, the TLS 1.3 presentation language, …
- **Protocol type systems**
  - eTPL, YANG, NetPDL, PADS, DataScript, PacketTypes, the Meta Packet Language, …

These languages couple external *and* internal representations: can't model protocols where these are different

Representing Protocol Data

- **Syntax description languages**
  - ABNF, ASN.1, the TLS 1.3 presentation language, …
- **Protocol type systems**
  - eTPL, YANG, NetPDL, PADS, DataScript, PacketTypes, the Meta Packet Language, …
- **Protocol representation systems**
  - Nail, Narcissus, …

Representing Protocol Data

- **Syntax description languages**
  - ABNF, ASN.1, the TLS 1.3 presentation language, …
- **Protocol type systems**
  - eTPL, YANG, NetPDL, PADS, DataScript, PacketTypes, the Meta Packet Language, …
- **Protocol representation systems**
  - Nail, Narcissus,

Need support for strong type guarantees *and* support for context-based, multi-stage parsing

We need a common representation
that is safe and extensible

The Network Packet Representation

- A typed protocol representation
- Decoupled from protocol description languages and target output languages
- Provides type constructors for a number of basic type classes, that can be composed into descriptions for complex protocols

The Network Packet Representation

```
An RTP Data Packet is formatted as follows:

 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                      [CSRC identifier list]                   |
|                         (4 * CC octets)                       |
|                          CC may be zero                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     defined by signalling     |    header extension length    | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                        header extension                       | | OPTIONAL
|                  format defined by signalling                 | | (if X=1)
|                                                               | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Payload                            |
|            (variable format and length, depends on PT)        |
|                                                               |
|               +-------........---------------+---------------+
|               |Padding (PadCnt octets, if P=1)|PadCnt (if P=1)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

where:

Version (V): 2 bits; V == 2.  This field identifies the version of RTP.
  The version defined by this specification is two (2).  (The value 1
  is used by the first draft version of RTP and the value 0 is used by
  the protocol initially implemented in the "vat" audio tool.)
```

The Network Packet Representation

```
An RTP Data Packet is formatted as follows:

 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                     [CSRC identifier list]                    |
|
|
+-
|
+-
|                       header extension                        | | OPTIONAL
|                 format defined by signalling                  | | (if X=1)
|                                                               | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Payload                            |
|            (variable format and length, depends on PT)        |
|                                                               |
|               +-------........---------------+---------------+
|               |Padding (PadCnt octets, if P=1)|PadCnt (if P=1)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

where:

Version (V): 2 bits; V == 2.  This field identifies the version of RTP.
  The version defined by this specification is two (2).  (The value 1
  is used by the first draft version of RTP and the value 0 is used by
  the protocol initially implemented in the "vat" audio tool.)
```

Bit strings to represent raw protocol data

The Network Packet Representation

```
An RTP Data Packet is formatted as follows:

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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     [CSRC identifier list]                    |
|                        (4 * CC octets)                        |
|                         CC may be zero                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      defined by signalling    |    header extension length    | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                       header extension                        | | OPTIONAL
|                 format defined by signalling                  | | (if X=1)
|                                                               | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Payload                            |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

where:

Version (V): 2 bits; V == 2.  This field identifies the version of RTP.
  The version defined by this specification is two (2).  (The value 1
  is used by the first draft version of RTP and the value 0 is used by
  the protocol initially implemented in the "vat" audio tool.)
```

Arrays to represent sequences of elements of the same type

The Network Packet Representation

```
An RTP Data Packet is formatted as follows:

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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                      [CSRC identifier list]                    
|                         (4 * CC octets)                       
|                          CC may be zero                       
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     defined by signalling     |    header extension length    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                       header extension                        | OPTIONAL
|                 format defined by signalling                  | (if X=1)
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Payload                              
|           (variable format and length, depends on PT)         
|                                                               
|                +-------........---------------+---------------
|                |Padding (PadCnt octets, if P=1)|PadCnt (if P=1)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

where:

Versio
  The v
  is us
  the p
```

Structures to represent packets themselves

The Network Packet Representation

```
An RTP Data Packet is formatted as follows:

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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                     [CSRC identifier list]                    |
|                        (4 * CC octets)                        |
|                         CC may be zero                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     defined by signalling     |   header extension length     | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                        header extension                       | | OPTIONAL
|                 format defined by signalling                  | | (if X=1)
|                                                               | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Payload                            |
|            (variable format and length, depends on PT)        |
```

Constraints within structures

```
where:

Version (V): 2 bits; V == 2.  This field identifies the version of RTP.
  The version defined by this specification is two (2).  (The value 1
  is used by the first draft version of RTP and the value 0 is used by
  the protocol initially implemented in the "vat" audio tool.)
```

The Network Packet Representation

```
An RTP Data Packet is formatted as follows:

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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                     [CSRC identifier list]                    |
|                        (4 * CC octets)                        |
|                         CC may be zero                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      defined by signalling    |   header extension length     | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                        header extension                       | | OPTIONAL
|                  format defined by signalling                 | | (if X=1)
|                                                               | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Payload                            |
|          (variable format and length, depends on PT)          |
```

Constraints within structures

```
where:

Version (V): 2 bits; V == 2.  This field identifies the version of RTP.
  The version defined by this specification is two (2).  (The value 1
  is used by the first draft version of RTP and the value 0 is used by
  the protocol initially implemented in the "vat" audio tool.)
```

The Network Packet Representation

```
An RTP Data Packet is formatted as follows:

 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                    [CSRC identifier list]                     |
|                       (4 * CC octets)                         |
|                        CC may be zero                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     defined by signalling     |    header extension length    | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                       header extension                        | | OPTIONAL
|                 format defined by signalling                  | | (if X=1)
|                                                               | |

|              +-------........---------------+---------------+
|              |Padding (PadCnt octets, if P=1)|PadCnt (if P=1)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

where:

Version (V): 2 bits; V == 2.  This field identifies the version of RTP.
  The version defined by this specification is two (2).  (The value 1
  is used by the first draft version of RTP and the value 0 is used by
  the protocol initially implemented in the "vat" audio tool.)
```

Contextual data shared out-of-band or between different PDUs

The Network Packet Representation

```
An RTP Data Packet is formatted as follows:

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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                     [CSRC identifier list]                    |
|                        (4 * CC octets)                        |
|                         CC may be zero                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      defined by signalling      |    header extension length    | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                       header extension                        | | OPTIONAL
```

A protocol is comprised of multiple PDUs

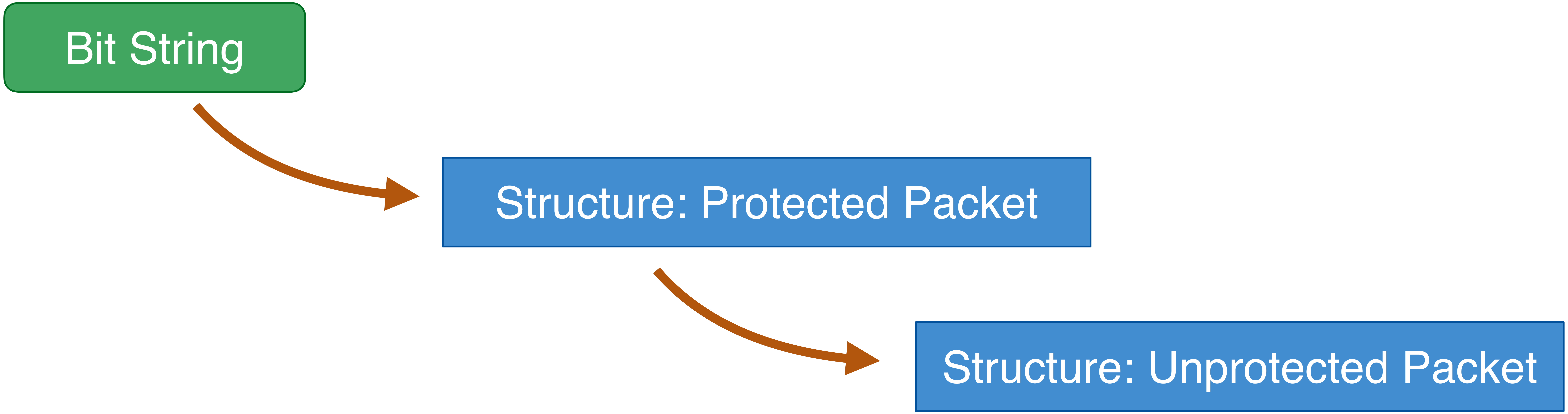
```
|           (variable format and length, depends on PT)         |
|                                                               |
|               +-------.........---------------+---------------+
|               |Padding (PadCnt octets, if P=1)|PadCnt (if P=1)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

where:

Version (V): 2 bits; V == 2.  This field identifies the version of RTP.
  The version defined by this specification is two (2).  (The value 1
  is used by the first draft version of RTP and the value 0 is used by
  the protocol initially implemented in the "vat" audio tool.)
```

Parsing Functions

- PDUs may have multi-stage parsing processes, with decryption or decompression necessary

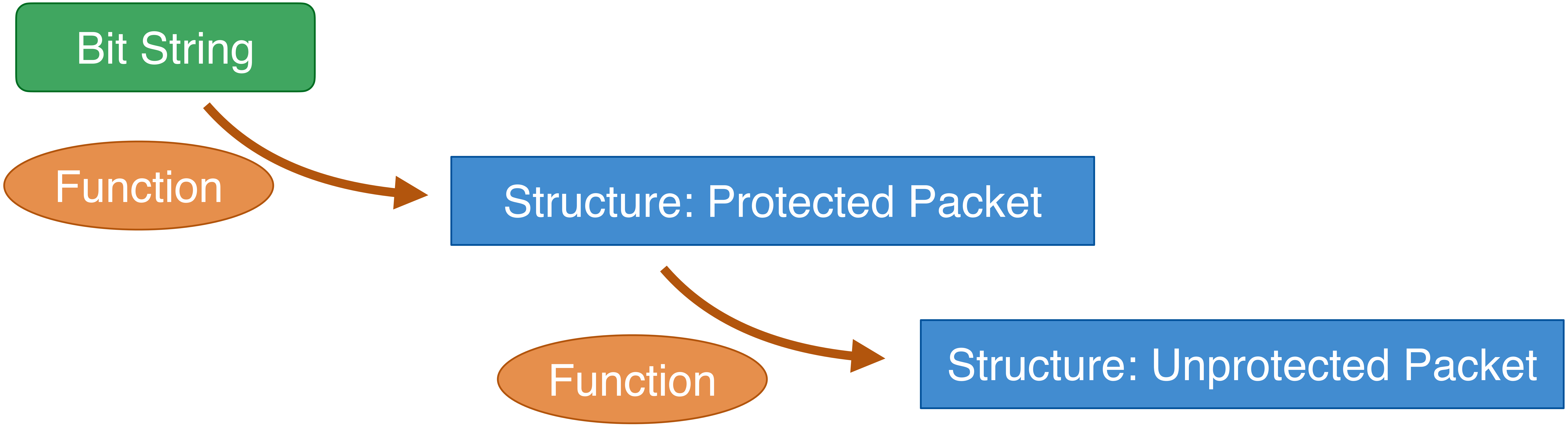
Parsing Functions

- PDUs may have multi-stage parsing processes, with decryption or decompression necessary



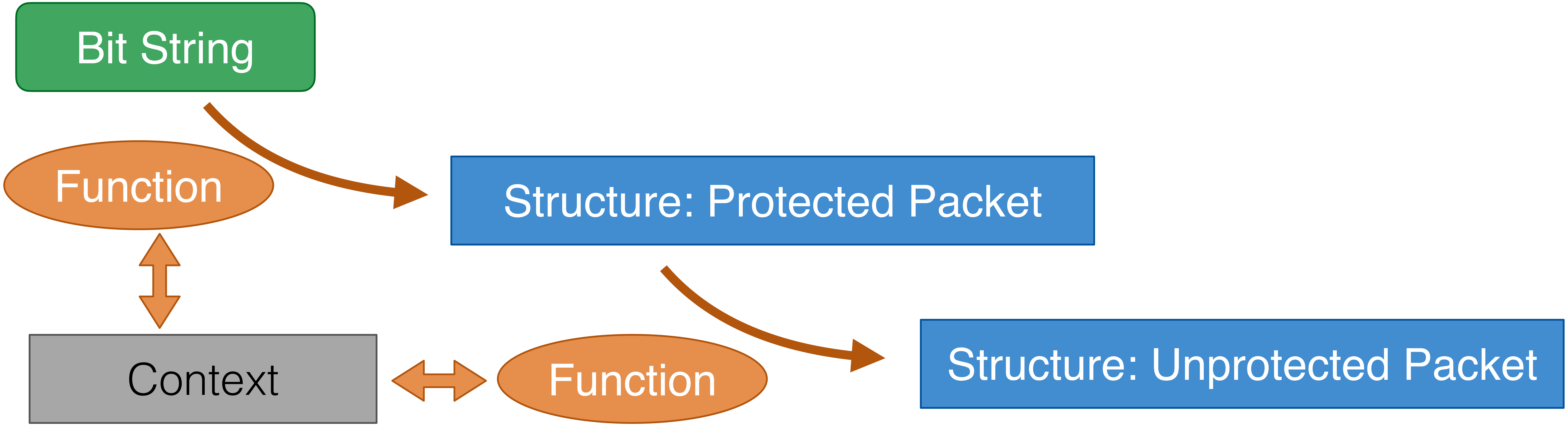
Parsing Functions

- PDUs may have multi-stage parsing processes, with decryption or decompression necessary



Parsing Functions

- PDUs may have multi-stage parsing processes, with decryption or decompression necessary



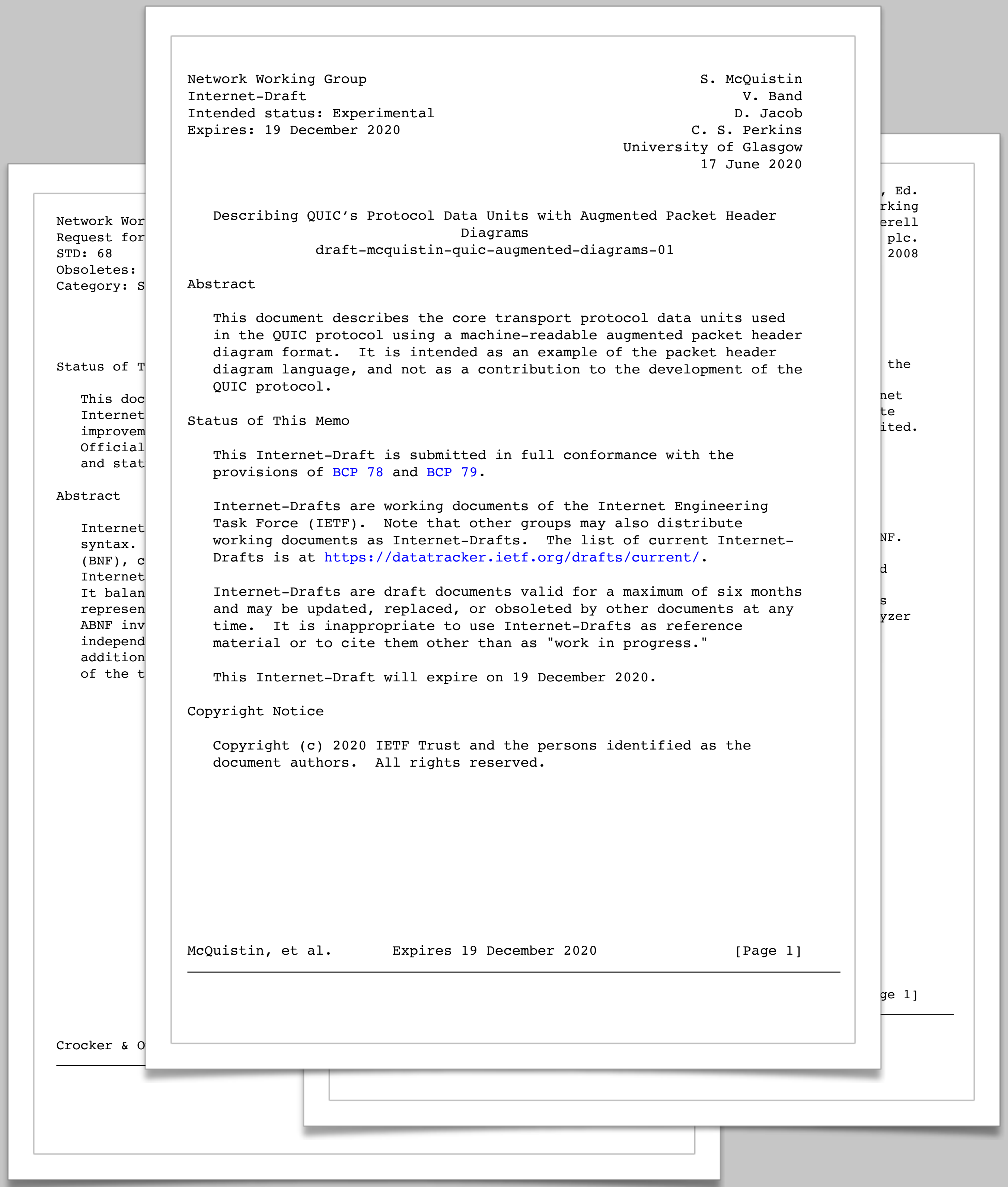
The Network Packet Representation

- A typed intermediate protocol representation, independent of input and output languages
- Enables state to be maintained between the parsing of different PDUs using typed *parsing contexts*
- Provides support for dependently formatted PDUs, constraints on and between PDU fields, and for multi-stage parsing via typed functions: all needed for parsing complex protocols

The Network Packet Representation

- A typed intermediate protocol representation, independent of input and output languages
- Enables state to be maintained between the parsing of different PDUs using typed *parsing contexts*
- Provides support for dependently formatted PDUs, constraints on and between PDU fields, and for multi-stage parsing via typed functions: all needed for parsing complex protocols

More details about the type system in the paper


Network Working Group S. McQuistin
Internet-Draft V. Band
Intended status: Experimental D. Jacob
Expires: 19 December 2020 C. S. Perkins
University of Glasgow
17 June 2020

Describing QUIC's Protocol Data Units with Augmented Packet Header Diagrams
draft-mcquistin-quic-augmented-diagrams-01

Abstract

This document describes the core transport protocol data units used in the QUIC protocol using a machine-readable augmented packet header diagram format. It is intended as an example of the packet header diagram language, and not as a contribution to the development of the QUIC protocol.

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 19 December 2020.

Copyright Notice

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

McQuistin, et al. Expires 19 December 2020 [Page 1]

Network Packet Representation

There are social barriers to the adoption of protocol description techniques

Integrating with Protocol Standards

- Most readers are human
- Authorship workflows are diverse
- Canonical specifications
- Expressiveness
- Minimise required change

Protocol Description Languages

- A wide number of languages are already in use: ABNF, ASN.1, YANG, the TLS 1.3 presentation language, …
- Any tool that aims to see broad adoption should accept multiple description formats
- The Network Packet Representation supports this: it is language agnostic
- Parsing structured description languages is well understood, and it should be possible to generate a Network Packet Representation from them
- Informal languages, like packet header diagrams, are more challenging

Augmented Packet Header Diagrams: QUIC example

```
An Initial Packet is formatted as follows:

  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
 +-+-+-+-+-+-+-+-+
 |1|1| T | R | P |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                            Version                            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    DCID Len   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                Destination Connection ID (DCID)             ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    SCID Len   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                 Source Connection ID (SCID)                 ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                         Token Length                        ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                            Token                            ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                            Length                           ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                         Packet Number                       ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                            Payload                          ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

where:

Header Form (HF): 1 bit; HF == 1.  The most significant bit (0x80) of
    byte 0 (the first byte) is set to 1 for long header packets.
```

Augmented Packet Header Diagrams: QUIC example

```
An Initial Packet is formatted as follows:

  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
 +-+-+-+-+-+-+-+-+
 |1|1| T | R | P |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                            Version                            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    DCID Len   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                Destination Connection ID (DCID)             ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    SCID Len   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                  Source Connection ID (SCID)                ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          Token Length                       ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                             Token                           ...
```

Maintains an easy-to-read diagram showing the layout of packets

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                            Payload                          ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

where:

Header Form (HF): 1 bit; HF == 1.  The most significant bit (0x80) of
    byte 0 (the first byte) is set to 1 for long header packets.
```

Augmented Pac ... UIC example

```
|1|1| T | R | P |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Version                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    DCID Len   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Destination Connection ID (DCID)              ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    SCID Len   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Source Connection ID (SCID)                 ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Token Length                        ...
```

Uses structured, but idiomatic, text to provide constraints and model parsing context use

```
|                            Payload                          ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

where:

Header Form (HF): 1 bit; HF == 1.  The most significant bit (0x80) of
    byte 0 (the first byte) is set to 1 for long header packets.

                                ...

DCID Len (DLen): 1 byte; DLen <= 20.  This field contains the length,
    in bytes, of the Destination Connection ID field that follows it.

Destination Connection ID (DCID): DLen bytes. The Destination
    Connection ID field is between 0 and 20 bytes in length.  On
    receipt, the value of DCID is stored as Initial DCID.

SCID Len (SLen): ...
```

Augmented Packet Header Diagrams: QUIC example

```
A Protected Packet is either a Protected Long Header Packet or a
Protected Short Header Packet.

An Unprotected Packet is either a Long Header Packet or a Short
Header Packet.

An Unprotected Packet is parsed from a Protected Packet using
the remove_protection function. The remove_protection function
is defined as:

func remove_protection(from: Protected Packet) -> Unprotected Packet:
   ...

An Unprotected Packet is serialised to a Protected Packet using
the apply_protection function. The apply_protection function is
defined as:

func apply_protection(to: Unprotected Packet) -> Protected Packet:
   ...
```

Augmented Packet Header Diagrams: QUIC example

A Protected Packet is either a Protected Long Header Packet or a

Provides support for functions and context use

An Unprotected Packet is parsed from a Protected Packet using
the remove_protection function. The remove_protection function
is defined as:

```
func remove_protection(from: Protected Packet) -> Unprotected Packet:
   ...
```

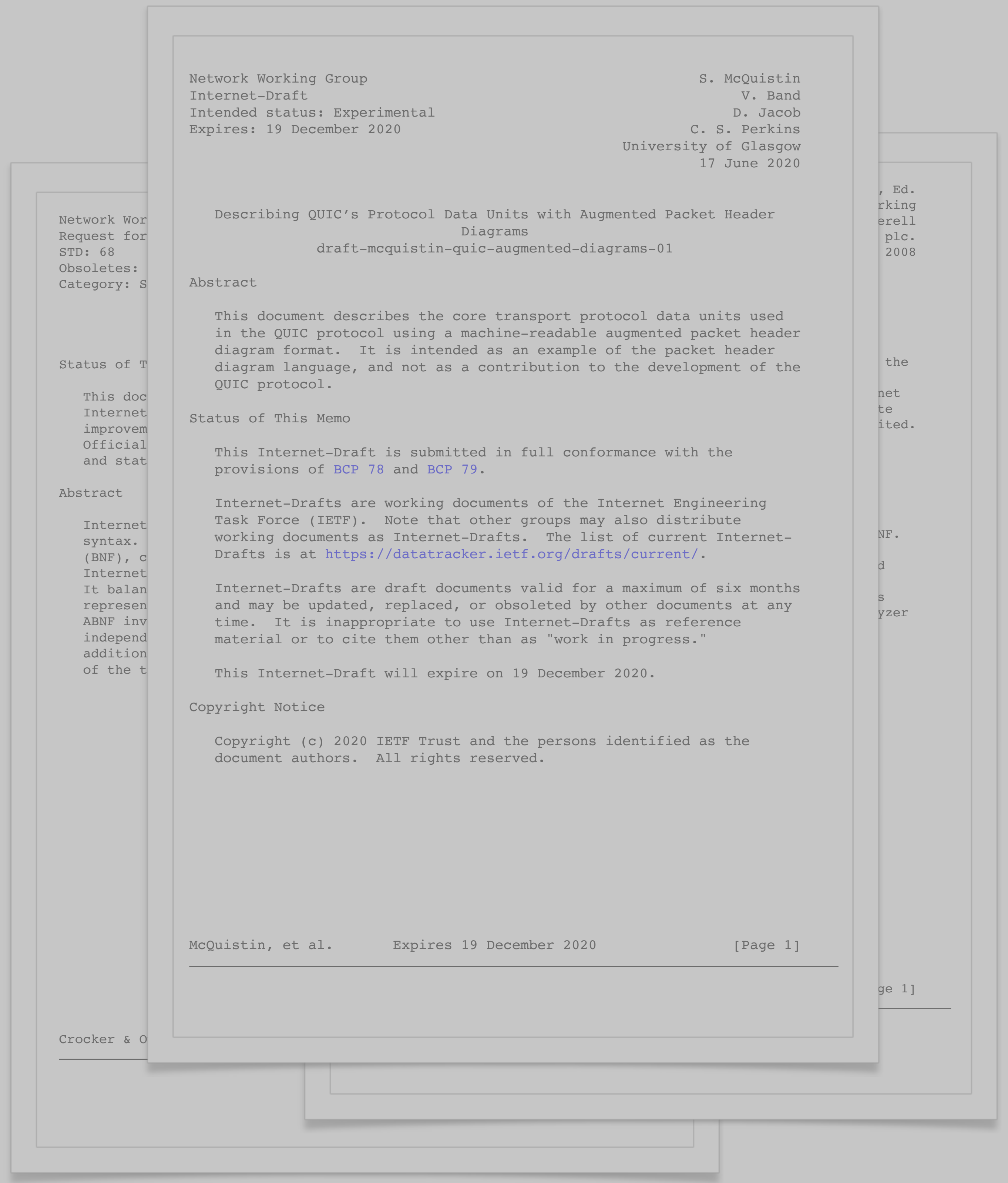
An Unprotected Packet is serialised to a Protected Packet using
the apply_protection function. The apply_protection function is
defined as:

```
func apply_protection(to: Unprotected Packet) -> Protected Packet:
   ...
```

Augmented Packet Header Diagrams

- The format of packet header diagrams can be regularised with minimal change
- The format remains extremely close to that in common use, easing adoption
- It balances structure and uniformity, needed for machine parsing, with the flexibility needed for practical use
- Prototype tooling that supports this input format, generating the Network Packet Representation from it

```
An Initial Packet is formatted as follows:

 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
+-+-+-+-+-+-+-+-+
|1|1| T | R | P |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Version                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   DCID Len    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Destination Connection ID (DCID)              ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   SCID Len    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Source Connection ID (SCID)                 ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Token Length                        ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Token                            ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Length                           ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Packet Number                       ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Payload                          ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

where:

Header Form (HF): 1 bit; HF == 1.  The most significant bit (0x80) of
    byte 0 (the first byte) is set to 1 for long header packets.

                                 ...

DCID Len (DLen): 1 byte; DLen <= 20.  This field contains the length,
    in bytes, of the Destination Connection ID field that follows it.

Destination Connection ID (DCID): DLen bytes. The Destination
    Connection ID field is between 0 and 20 bytes in length.  On
    receipt, the value of DCID is stored as Initial DCID.

SCID Len (SLen): ...
```


Network Working Group S. McQuistin
Internet-Draft V. Band
Intended status: Experimental D. Jacob
Expires: 19 December 2020 C. S. Perkins
University of Glasgow
17 June 2020

Describing QUIC's Protocol Data Units with Augmented Packet Header Diagrams
draft-mcquistin-quic-augmented-diagrams-01

Abstract

This document describes the core transport protocol data units used in the QUIC protocol using a machine-readable augmented packet header diagram format. It is intended as an example of the packet header diagram language, and not as a contribution to the development of the QUIC protocol.

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 19 December 2020.

Copyright Notice

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

McQuistin, et al. Expires 19 December 2020 [Page 1]

Network Packet Representation

Automatic parser generation provides a number of opportunities to improve security

Parser Generators

- The Network Packet Representation can be used to generate implementation code in any number of target programming languages
- Core code generation functions can be implemented once, easing the development of code generators for new languages

QUIC example

An Initial Packet is formatted as follows:

```
 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
+-+-+-+-+-+-+-+-+
|1|1| T | R | P |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Version                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   DCID Len    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Destination Connection ID (DCID)            ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   SCID Len    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Source Connection ID (SCID)               ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Token Length                       ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Token                           ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Length                          ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Packet Number                      ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Payload                         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

Header Form (HF): 1 bit: HF == 1. The most significant bit (0x80) of

QUIC example

```
An Initial Packet is formatted as follows:

 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
+-+-+-+-+-+-+-+-+
|1|1| T | R | P |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Version                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   DCID Len    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Destination Connection ID (DCID)              ...
                                                          +-+-+-+

                                                          +-+-+-+
|                 Source Connection ID (SCID)                 ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Token Length                        ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Token                            ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Length                           ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Packet Number                       ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Payload                           ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

where:

Header Form (HF): 1 bit: HF == 1.  The most significant bit (0x80) of
```

Emit types and parser combinator functions for each field type

QUIC example

Emit types and parser combinator functions for structures

```
An Initial Packet

 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
+-+-+-+-+-+-+-+-+
|1|1| T | R | P |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Version                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   DCID Len    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Destination Connection ID (DCID)              ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   SCID Len    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Source Connection ID (SCID)                  ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Token Length                         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Token                             ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Length                            ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Packet Number                        ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Payload                            ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

where:

Header Form (HF): 1 bit: HF == 1   The most significant bit (0x80) of
```

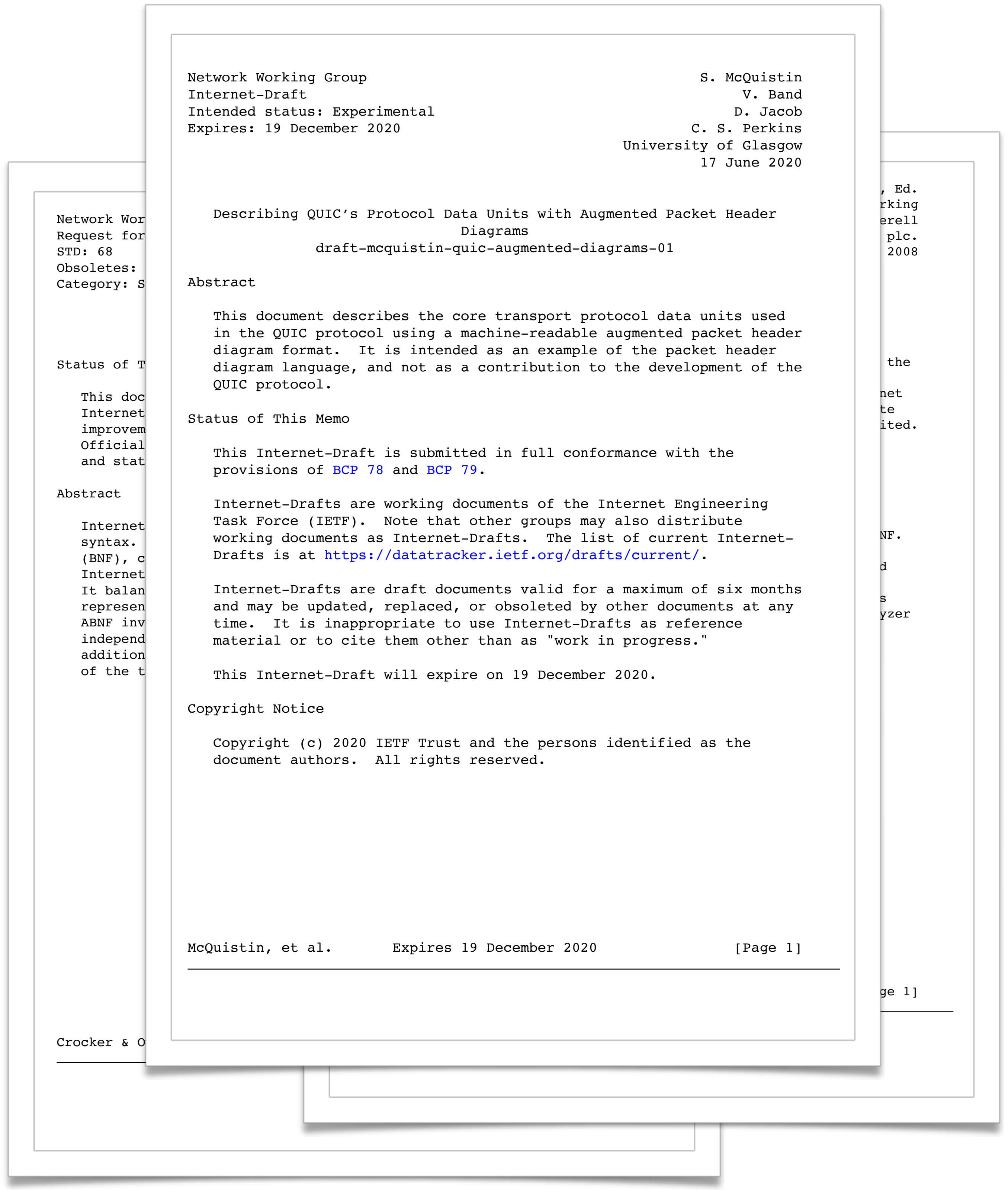
QUIC example

```
A Protected Packet is either a Protected Long Header Packet or a
Protected Short Header Packet.

An Unprotected Packet is either a Long Header Packet or a Short
Header Packet.

An Unprotected Packet is parsed from a Protected Packet using
the remove_protection function. The remove_protection function
is defined as:

func remove_protection(from: Protected Packet) -> Unprotected Packet:
   ...

An Unprotected Packet is serialised to a Protected Packet using
the apply_protection function. The apply_protection function is
defined as:
```

Generate stubs for functions

Parser Generators

- Support for different parser models — like parser combinators — can be implemented once
- This has implications for security: modern systems languages, like Rust, can be easily supported, encouraging their adoption and use
- Our prototype tooling supports Rust code generation



```
Network Working Group                                        S. McQuistin
Internet-Draft                                                    V. Band
Intended status: Experimental                                    D. Jacob
Expires: 19 December 2020                                    C. S. Perkins
                                                  University of Glasgow
                                                            17 June 2020


   Describing QUIC's Protocol Data Units with Augmented Packet Header
                               Diagrams
             draft-mcquistin-quic-augmented-diagrams-01

Abstract

   This document describes the core transport protocol data units used
   in the QUIC protocol using a machine-readable augmented packet header
   diagram format.  It is intended as an example of the packet header
   diagram language, and not as a contribution to the development of the
   QUIC protocol.

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 19 December 2020.

Copyright Notice

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



McQuistin, et al.        Expires 19 December 2020               [Page 1]
```

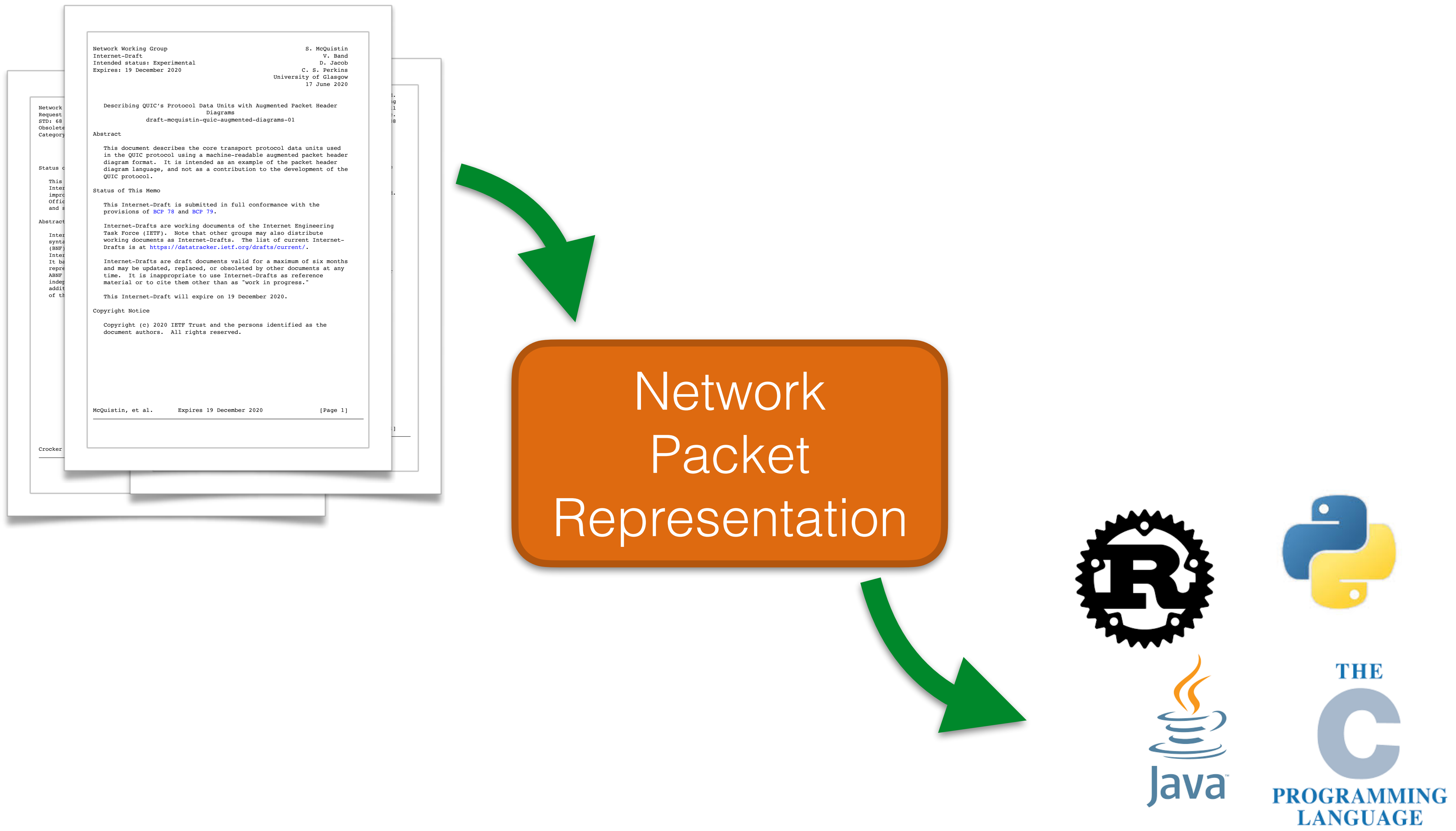
Network
Packet
Representation

Java

THE C PROGRAMMING LANGUAGE

Conclusions

- Support for complex protocols with contextual, multi-stage parsing processes
- An incremental path to adoption within the standards community
- An important step towards the routine use of parser generating tooling, that should lead to standards that are safer and more trustworthy



Paper: https://irtf.org/anrw/2020/program.html#p21