Bits

QUIC Interim, New York, September 2018

Martin Thomson

Goal

Resolve what to do with the first octet

Assume that the spin bit discussion will result in

1. no use of bits
2. a single bit (spin)
3. 3 bits (spin+VEC)

...and that no other outcome is possible

Plan for all possible outcomes:

discuss proposals, suggest amendments, agree in principle

Constraints Review: RFC 7983 Mux

0-3	STUN
16-19	ZRTP
20-63	DTLS
64-79	TURN Channel
128-191	SRTP

High Priority

STUN: `0b0000.00xx` (0-3)

Desirable

SRTP: `0b10xx.xxxx` (192-255)

DTLS: `0b0001.01xx`, `0b0001.1xxx`, `0b001x.xxxx` (20-63)

Take it or leave it

TURN Channels: `0b0100.xxxx` (64-79)

ZRTP: `0b0001.00xx` (16-19)

Constraints (and Wants) Review

Type in the clear (it determines key)

Packet number **and key phase** encrypted

Greasing

- Either we use the bits or they will gain new meaning
- Both use and encrypt is best, but either alone might work
- Fixed values are worst, but might be necessary

Proposal: Common Features

Move packet number length encoding into the first octet

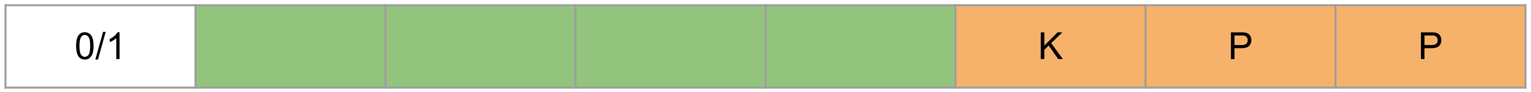
- 2 octets can represent 1-4 octets of packet number

Put it alongside key phase

Encrypt both

Requires packet number protection key to not change with key update (simple change)

That means that we have 4 bits free



Proposal: Long Header

No spin bits here, so one proposal*:

1	T	T	T	T	K	P	P

Problem: some unused values (or 4 codepoints for each type)

We only need two bits for type so we might encrypt some zero/randomized bits instead:

1	T	T	Z/R	Z/R	K	P	P

Suggestion: adjust to match the short header decision

Long Header Types

A. Peer-to-peer probably won’t need Retry and 0-RTT

   Avoid SRTP by reallocating Retry and 0-RTT so that they conflict with `0b`**`10`**`xx.xxxx`

B. #1655 proposes simpler identification of Initial and 0-RTT

   That proposes putting Initial and 0-RTT on `0b`**`11`**`xx.xxxx` so that it is easier to identify packets with server-chosen connection IDs (`packet[0] < 0xfe`)

But wait! `packet[0] < 0xfe` doesn’t work now anyway

Proposal: Long Header Types

Retry:	0b1**01**x.xxxx
0-RTT:	0b1**00**x.xxxx
Initial:	0b1**10**x.xxxx
Handshake:	0b1**11**x.xxxx

```
define hasServerChosenCid(packet):
   return packet[0] & 0xa0 != 0x80
```

Short Header with 3 Spin Bits (Spin+VEC)

Proposal: Fix one bit and avoid STUN, DTLS, and ZRTP:

0	1	S	VEC	VEC	K	P	P

That is restrict this to the range from 64-255

Risk: 0-63 could be unusable for QUICv2

Risk: unused values for type in long header if this matches

Suggestion: allocate 4 codepoints for each type

Proposal: Long Header Types (4-bit version)

Retry and 0-RTT: `0b10aa.axxx`

… with **`aaa`** shuffled and split between the two

Initial and Handshake: `0b11bb.bxxx`

… with **`bbb`** shuffled and split between the two

Short Header with 1 Spin Bit (No VEC)

Same design as previous, but with more bits encrypted

0	1	S	Z/R	Z/R	K	P	P

If we cared about TURN channels we could do this

0	1	1	S	Z/R	K	P	P

Short Header with 1 Spin Bit (No VEC)

Same design as previous, but with more bits encrypted

0	1	S	Z/R	Z/R	K	P	P

If we cared about TURN channels we could do this

~~0~~	~~1~~	~~1~~		~~Z/R~~	~~K~~	~~P~~	~~P~~

I haven’t heard from anyone that cares about TURN channels

Short Header with No Spin Bit

Sure, we could encrypt another bit, but then short and long can’t be the same, which is annoying, and a separate greasing mechanism is also annoying

So maybe TURN channels can get a pass

0	1	1	Z/R	Z/R	K	P	P

Cost here is more exposure to ossification (0-95)