Header Compression for TLV-based Packets

ICNRG Yokohama

Marc Mosko

Nov 5, 2015

“Header Compression” in TLV World

- Compress all the signaling
  - Fixed Header
  - T and L fields
  - V fields except user payload
    - KeyId
    - Public Keys
    - Name Components
    - Timestamps
    - Anything that is predictable

Motivation for something new

- Network packets are small
  - Gzip, bzip2, etc. usually expand packet because of their block encoding structure.
  - Microsoft point-to-point compress (MPPC, RFC 2118) only has minor savings, sometimes bigger.
- Dictionary and window algorithms
  - Require state exchange, lost packets result in burst errors or decoding delay.
  - Need a lot of buffer space if there are packets from mixed flows.

Why is gzip bad?

- 10 byte header, 3 byte footer.
- Back references are 3 bytes, minimum
  - But repeating T values are 2 bytes.
  - Exact patterns do not repeat much, but some fields have high redundancy that we can remove with context-dependent substitutions.
  - Won't even work for 1/3/5 encoding with 1+1
- It will build up many short dictionary entries on cryptographic fields.
- It has to transmit the dictionary.

Why is bzip2 bad?

- Run-length encoding of 4+ byte too long
- 100k – 900k block size
- 4-byte header, 4-byte footer
- 20+ byte block header

What about window/learning

- OK between two consistent peers
  - 1-hop peer ok.
  - Otherwise, Interests can go anywhere unless you use topological name.
- Losses cause burst errors unless use ACKS
  - Leads to delay in using learned values.
  - Tradeoff between loss and burst errors.
- ICN packets might be very large
  - Need large history window, so finding longest string match might be pretty expensive.

Example (interest)

- Interest with fixed header and 2+2 TLV

  /bell/0x01020304/0x05060708/0x090a0b0c

Method	Bytes
Data (name)	**16**
Uncompressed	**48**
gzip -9	**77**
bzip2 -9	**75**
MPPC (RFC 2118)	**42**
TLV compression	**28**

Example (Content Object)

- Content object w/ 162-byte public key, 32-byte keyid, and 128-byte signature, etc.

Method	Bytes
Data (name, payload, pubkey, keyid, sig)	**372**
Uncompressed	**436**
gzip -9	**461**
bzip2 -9	**574**
MPPC (RFC 2118)	**448**
TLV compression	**396**

Overview

- Static TL compression
  - Allows reducing the overhead caused by TL encoding (2+2 and 1/3/5) *without state exchange*.
- Dictionary learned replacement
  - Learn strings like Key IDs and Public Keys. Those are long random byte strings.
  - Use delta encoding for things like Chunks or times or serial numbers.
- Byte-aligned on 'T' boundaries.

Outline of Algorithm

- Fixed header has a “compressed” flag
  - Version field is only 4 bits
  - If not set, uses 8 byte FH and 2+2 TLs
  - If set,
    - 1-byte context header (2bit flats, 3bit CID, 3bit CRC)
    - use 3, 4, or 8 byte FH and 1 – 5 byte TLs
- In “compressed” mode
  - Static TL pair or (TL)*TL string (in to 1 byte)
  - Static T, variable L (in to 1, 2, 3, 4 or 5 bytes)
  - Learned TLV replacement (in to 2, 3, or 4 bytes)
  - Learned TLV counter (only send offset from base)

Initialization

- Before using compression
  - Peers exchange willingness to compress.
  - Peers exchange capabilities
    - Maximum buffer size (used for window based dictionary definitions).
    - Name of static dictionary used, if not the default.
  - If using non-standard static dictionary
    - Exchange the dictionaries.
  - Done at link initialization or with in-band link management.
  - Determine a Context ID (CID) for this state.

State Exchange

- Out-of-band
  - Use a separate packet with FixedHeader PacketType = Dictionary
  - Sends one or more definitions.
  - Has Seqnum for reliable state exchange.
- In-band
  - Footer sends dictionary definitions, using (backwards_offset, length) back in to the packet.
  - Carries seqnum for reliable state exchange.
  - Has own CRC
- State exchange ACK

TL values

- CCNx 1.0
  - Re-uses "T" values as it's context dependent. So, very few actual "T" values. Leads to highly-compressable packet format.
- NDN 1/3/5
  - Uses a global "T" space. Use a pre-processor to map common values in context to high-redundancy values.

Entropy examples

- Based on random source model for an Interest.
- TL + V uses 6-component name with 5 repeated.

	H (bit-aligned)	H (byte-aligned)	2+2	1/3/5	TL comp.
TL only	4.9	8.0	32.0	18.9	8.0
TL + V	8.4	11.7	88.3	55.4	14.8

Conclusion

- Initialization stage
  - Use static dictionary to compress TLs.
  - Compress fixed header.
  - Can be used inside encryption envelope too.
- Learning stage
  - Use reliable state exchange to compress TLVs.
  - TLV pattern substitution.
  - Counter type for delta encoding.
- Have running code (python) for static dictionary