rohc
Robust Header Compression

52nd IETF December 2001
Salt Lake City

Chairs:

Carsten Bormann <cabo@tzi.org>
Mikael Degermark <micke@cs.arizona.edu>

Mailing List:

rohc@cdt.luth.se

52nd IETF: Agenda (from 30000 feet)

- 1. WG chair admonishments
- 2. Real agenda

✓Blue sheets
✓Scribe

Hello! This is an IETF Working Group

- We are here to make the Internet work (Fred Baker)
  - Together! (Harald Alvestrand)
- Rough Consensus and Running Code (Dave Clark)
- Working Group is controlled by
  - IETF Process (RFC2026, RFC2418) – *read it!*
  - Area Directors (ADs): Alison Mankin, Scott Bradner
  - Charter (http://www.ietf.org/html.charters/rohc-charter.html) -- *read it!*
  - Working Group Chairs: Mikael Degermark, Carsten Bormann
  - Technical Advisor: Erik Nordmark
- Work is done on email list rohc@cdt.luth.se
  - And on IETF meetings, interim meetings, informal meetings, …
  - Mailing list is official channel, though

RFC 2026: Internet Standards Process

- **Standards track RFCs:**
  - **WG consensus (as judged by WG chairs)**
  - **WG last call**
  - **IESG approval (based on AD recommendation)**
    - **Quality control!**
  - **IETF last call**
- **Informational RFCs**
- **BCP (best current practice) RFCs**

RFC 2026: IPR issues (1)

- (10.2) No contribution that is subject to any requirement of confidentiality or any restriction on its dissemination may be considered […]
- Where the IESG knows of rights or claimed rights […] the IETF Executive Director shall attempt to obtain from the claimant […] a written assurance that upon approval by the IESG of the relevant Internet standards track specification(s), any party will be able to obtain the right to implement, use and distribute the technology […] based upon the specific specification(s) under openly specified, reasonable, non-discriminatory terms.

RFC 2026: IPR issues (2)

- **Contributions (10.3.1(6)):**
  "The contributor represents that he has disclosed the existence of any proprietary or intellectual property rights in the contribution that are reasonably and personally known to the contributor."
- **I.e., if you know of a patent application for a technology you are contributing, you have to tell.**
  Or just shut up entirely!

ROHC: Charter (4) Goals and Milestones

Done
in last-call
Working
To do

- Mar: I-D on Requirements for IP/UDP/RTP HC.
- May: I-D of layer-2 design guidelines.
- May: I-D(s) proposing IP/UDP/RTP HC schemes.
- May: I-D of Requirements for IP/TCP HC.
- Jun: Requirements for IP/UDP/RTP HC submitted to IESG (Inf.)
- Jul: Requirements for IP/TCP HC submitted to IESG (Inf.)
- Jul: Resolve possibly multiple IP/UDP/RTP HC schemes into a single scheme.
- Aug: I-D on IP/TCP header compression scheme.
- Sep: Layer-2 design guidelines submitted to IESG (Inf.) ➡ TCP g/l
- Sep: IP/UDP/RTP HC scheme submitted to IESG (PS)
- Dec: IP/TCP HC scheme submitted to IESG (PS)
- Jan: Possible recharter of WG to develop additional HC schemes.

52nd IETF: Agenda (Thu morning)

- 0900 Chair admonishments and agenda (10)
- 0910 WG status & AD address (10)
- 0920 WG document status (10)
- 0930 Input from ROHC in the desert
  - 0930 Results from Tucson Kremer (10)
  - 0940 Discussion/Implications (10)
- 0950 Signaling compression
  - 0950 Overview Bormann (10)
  - 1000 Universal Decoder – workable? Price/Hannu (45)
  - 1045 Protocol: basic, extended Hannu/Price (15)
  - 1100 IPR Strategy (10)
  - 1110 Requirements met? (10)
  - 1120 Security issues (10)

52nd IETF: Agenda (Thu afternoon)

- 1530 TCP
  - 1530 TCP field behavior — West (10)
  - 1540 Requirements document ➡ freeze? — Jonsson (10)
  - 1550 Role of EPIC — West (10)
  - 1600 Progress in merging the drafts — (Authors) (10)
  - 1610 Requirements unmet so far — Jonsson (10)
- 1620 SCTP — Schmidt (20)
- 1640 MIB — Quittek (20)
- 1700 RTP ➡ DS — Chairs (10)
- 1710 Rechartering — (20)

Document status: WG RFCs: RTP ROHC

- **Published:**
  - **RFC3095: Framework and four profiles (was: draft-ietf-rohc-rtp-09.txt)**
  - **RFC3096: RTP requirements (was: draft-ietf-rohc-rtp-requirements-05.txt)**
- **Already part of 3GPP Release 4**
  - **Alongside with R99's inclusion of RFC2507 (*not* RFC2508!)**
- **Adopted by 3GPP2**
  - **Release C end 2001**

Document status: Lower layer guidelines

- draft-ietf-rohc-rtp-lower-layer-guidelines-01.txt
  Completed WG last-call in December 2000
- draft-ietf-rohc-rtp-lower-layer-guidelines-03.txt
  Prescriptive text changed to descriptive text
- One more editorial round ➡ second last-call 2001-12

Document status: ROHC over PPP

- draft-ietf-rohc-over-ppp-03.txt
  Completed last-call 2001-08-31
- draft-ietf-rohc-over-ppp-04.txt
  has the resulting clarifications (thanks, Lars-Erik)
- Submitted to the IESG on 2001-11-17 for PS

Document status: LLA ROHC

- draft-ietf-rohc-rtp-0-byte-requirements-01.txt,
  draft-ietf-rohc-rtp-lla-00.txt
  Completed last-call 2001-08-30
- Revised (no NHP for R-mode)
  draft-ietf-rohc-rtp-lla-03.txt
  Completed last-call 2001-12-06 (no comments)
- Submitted to IESG on 2001-12-06 for Info, PS

- Work on R-mode for LLA continues

52nd IETF: Agenda (Thu morning)

- 0900 Chair admonishments and agenda (10)
- 0910 WG status & AD address (10)
- 0920 WG document status (10)
- 0930 Input from ROHC in the desert
  - 0930 Results from Tucson Kremer (10)
  - 0940 Discussion/Implications (10)
- 0950 Signaling compression
  - 0950 Overview Bormann (10)
  - 1000 Universal Decoder – workable? Price/Hannu (45)
  - 1045 Protocol: basic, extended Hannu/Price (15)
  - 1100 IPR Strategy (10)
  - 1110 Requirements met? (10)
  - 1120 Security issues (10)

ERICSSON

ROHC Interop test II
November 2001, Tucson

Péter Krémer

Peter.Kremer@ericsson.com

Ericsson Research, Conformance Lab

ROHC Interop test II

- **Particulars:**
  - Place: Tucson, AZ
  - Date: 13-20 November, 2001
  - Host: Effnet
- **Participants:**
  - Effnet
  - Siemens/Roke Manor
  - Ericsson
  - (Nokia)
- **Support:**
  - Universität Bremen
  - University of Arizona

What did we test?

- ROHC over PPP
  - without negotiation
- ROHC over UDP
- Profile 1 (IP/UDP/RTP), IPv4
  - basic communication (mode transitions, new CRC),
  - change in incoming stream (TOS, TTL, IPID, TS_STRIDE),
  - robustness (packet loss, bit errors)
- Profile 0 (uncompressed), IPv4
  - basic communication

Clarifications needed

- IP-ID (original value in IR, IR-DYN <--> compressed value otherwise)
- Extension-3 in UO-1 packets
- Multiple sequence number options in one feedback packet
- Reparsing UOR-2 packet when flags changed in Extension-3
- Different types of ACKs during mode transition
- How?
  - Update Implementer's guide (draft-kremer-rohc-impguide-00.txt)
  - comments and all inputs are welcome

Next Interop

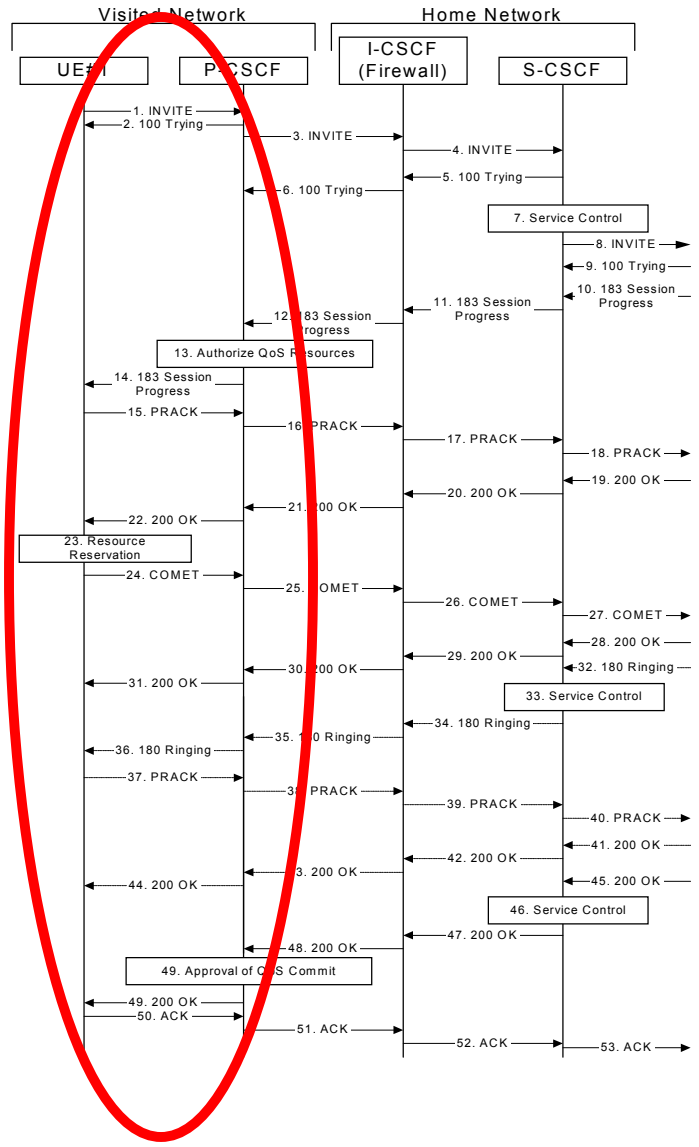
- Ericsson invites everybody
- Date (April?)
- Focus
  - all four profiles
  - robustness
  - IPv6
  - list-based compression
  - ROHC over PPP (with negotiation)
- Details will come on the mailing list

52nd IETF: Agenda (Thu morning)

- 0900 Chair admonishments and agenda (10)
- 0910 WG status & AD address (10)
- 0920 WG document status (10)
- 0930 Input from ROHC in the desert
  - 0930 Results from Tucson Kremer (10)
  - 0940 Discussion/Implications (10)
- 0950 Signaling compression
  - 0950 Overview Bormann (10)
  - 1000 Universal Decoder – workable? Price/Hannu (45)
  - 1045 Protocol: basic, extended Hannu/Price (15)
  - 1100 IPR Strategy (10)
  - 1110 Requirements met? (10)
  - 1120 Security issues (10)

Why?

- Minimize **connection setup delay** in complex 3GPP SIP interactions
- Minimize **bandwidth stealing** for in-call usage of SIP
- The point is *not* saving raw bandwidth (although it does help the network!)
- **draft-ietf-rohc-signaling-req-assump-03.txt**



What are the messages to be compressed?

- SIP:
  - Largely a lock-step protocol
  - Essentially RFC822 (Text)
  - Can carry MIME payload
- SDP:
  - v=2 m=audio etc. (Text)
  - Other MIME payloads are possible (SDPng!)
- Either *could* be encrypted, possibly partially

- RTSP (for streaming), also carrying SDP
- DNS, RSVP, … ???

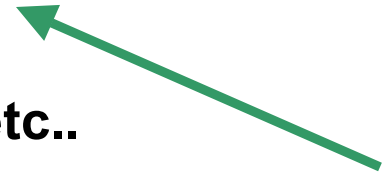
Signaling compression in ROHC WG – status

- Will be in next version of charter
- Highly sought after by 3GPP (for R5)
  - Not much time left!
- Useful for other signaling over low-bandwidth links
  - Applications in instant messaging?

- WG documents:
  - draft-ietf-rohc-signaling-req-assump-03.txt
  - draft-ietf-rohc-sigcomp-02.txt
  - draft-ietf-rohc-sigcomp-algorithm-00.txt

Signaling Compression: Components

- 1) The protocol
  - Message handling,
    - E.g. Verification of correct decompression
    - E.g. Usage of previous messages in the compression process
    - E.g. Context state handling (dictionary/codebook handling), excluding algorithm-specific aspects
- 2) The actual Compression Algorithm
  - What to save in the dictionaries/codebooks etc..
  - Compressed message representation
    - E.g. Lempel-Ziv based representations
  - IPR rathole



Movable boundary

Universal decompressor

- Hard to decide on a standard default algorithm
- Why not have the compressor tell the decompressor?
  - But avoid gazillion of incompatible registrations
- Universal Decompressor
  - Virtual machine optimized for decompression
  - Gets executable decompressor spec from compressor
  - No compression schemes in standards
  - Full interoperability with any compressor

- draft-ietf-rohc-sigcomp-algorithm-00.txt

Minimal Protocol

- UDP: per-packet, TCP: per-stream compression
- Start out with state reference
  - Decompressor spec
  - Initial dictionary
- Can use implicit ACK to ascertain that state is there
  - Loading dictionary with INVITE is likely good enough
- Extended versions can use explicit ACKs and compressor-decompressor state sharing
  - IPR issues
- draft-ietf-rohc-sigcomp-02.txt

Security requirements

- Secure state referencing
  - Avoid snooping into state of other users
  - Avoid unauthorized changes to state
- DoS vulnerabilities
  - Can't use decompressor as amplifier
  - Can't DoS-attack the decompressor by filling it with state
- Halting Problem
  - Limit number of VM instructions per packet
  - Make looping primitive consume input (indirect limit)

52nd IETF: Agenda (Thu morning)

- 0900 Chair admonishments and agenda (10)
- 0910 WG status & AD address (10)
- 0920 WG document status (10)
- 0930 Input from ROHC in the desert
  - 0930 Results from Tucson Kremer (10)
  - 0940 Discussion/Implications (10)
- 0950 Signaling compression
  - 0950 Overview Bormann (10)
  - 1000 Universal Decoder – workable? Price/Hannu (45)
  - 1045 Protocol: basic, extended Hannu/Price (15)
  - 1100 IPR Strategy (10)
  - 1110 Requirements met? (10)
  - 1120 Security issues (10)

SIEMENS

Roke Manor Research

Universal Decompressor

Richard Price

Universal Decompressor

- Similar in concept to a Java Virtual Machine
  - Optimised specifically for decompression algorithms
- Algorithms can be downloaded in three ways
  - Appended to front of first compressed message
  - Standalone packet before first compressed message
  - During negotiation of compression scheme
- Mnemonic language is provided

```
:next_character

HUFFMAN ($compressed_pointer, $bit_offset, position, 1, 16, 0)
HUFFMAN ($compressed_pointer, $bit_offset, length, 1, 16, 0)
COPY-LITERAL ($position, $length, $uncompressed_end)
COMPARE ($compressed_pointer, $compressed_end, next_character, 0, 0)
```

Why a Universal Decompressor?

- Why not negotiate the compression scheme?
  - Tough to pick a mandatory default algorithm
  - SIP-specific algorithms: not future-proof
  - Generic algorithms: high overhead
  - Hybrid algorithms: complex
- Why not use a Java Virtual Machine?
  - Processing and memory should be low compared to compression algorithm
  - Typical algorithm requires 8K working memory

How Universal is “Universal”?

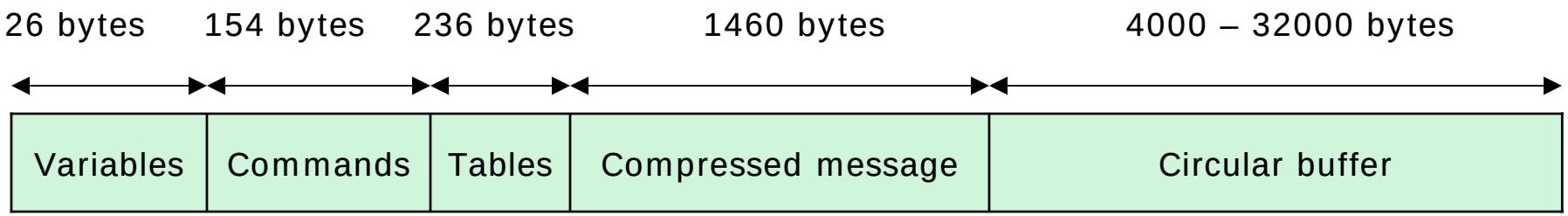
- Theory:
  - Universal decompressor is Turing-complete
  - Arbitrary decompression algorithms can be supported (given enough processing and memory)
- Practice:
  - Proven support for LZ77-based, LZ78-based and SIP-aware algorithms
  - LZ77, LZSS, LZW, DEFLATE, LZJH, EPIC

Processing and Memory Requirements

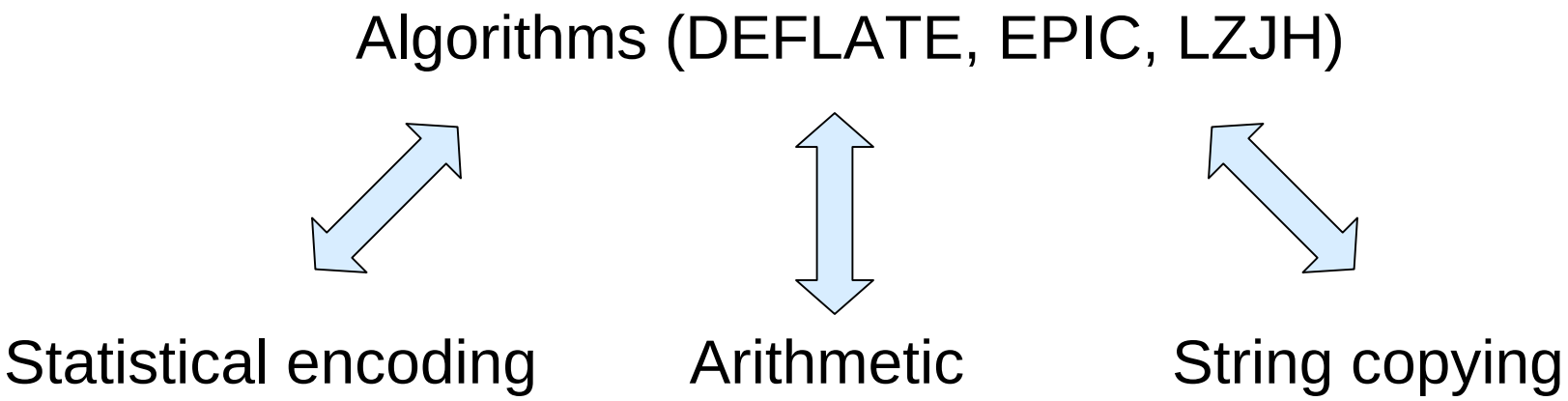
- Code size for proposed algorithms is small

Algorithm	Code size (bytes)	Commands per character
Simple LZ77	96	4
LZSS	99	7
LZW	132	8
DEFLATE (RFC 1951)	390	4 or 13
LZJH	313	7 or 11
EPIC	Depends on BNF	3 or 4

- Compares favourably to overall memory requirements

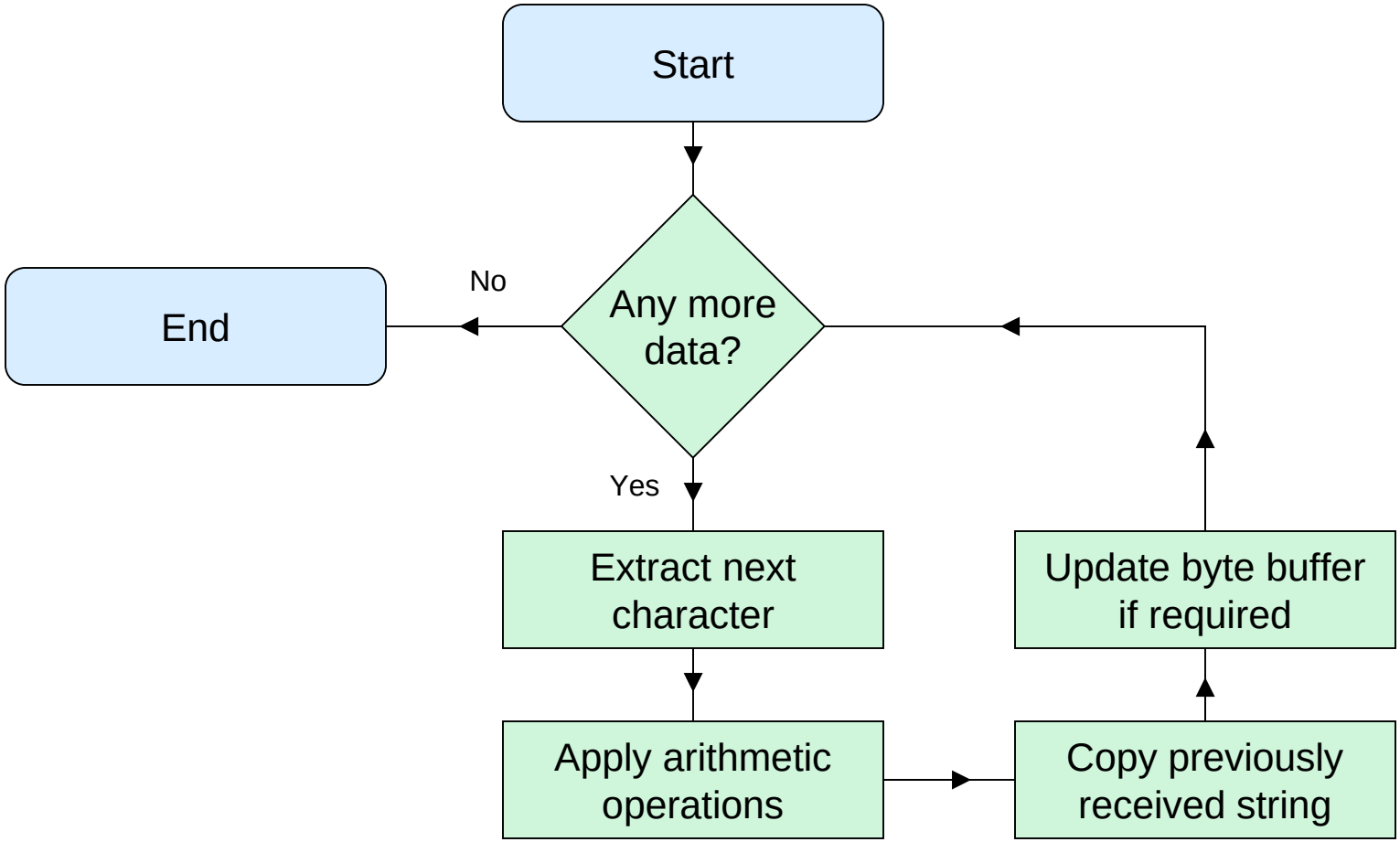


Choosing the Instructions



- Statistical encoding maps uncompressed integers to compressed bit patterns
- Arithmetic operations pre-process the uncompressed data to improve the compression ratio
- String copying replaces a string with a reference

Typical Decompression Algorithm



Available Instructions

Type	Instructions	Purpose
Statistical	HUFFMAN	Extracts characters from compressed data
Arithmetic	ADD	Modifies uncompressed character (e.g. to become a codebook reference)
	SUBTRACT	
	MULTIPLY	
	DIVIDE	
String copying	COPY	Copies previously received data
	COPY-LITERAL	
	COPY-OFFSET	
Program flow	SWITCH	Alters program execution
	COMPARE	
	CALL ... RETURN	

Open Issues

- Do we need to add additional instructions?
  - Bit manipulation operations
  - Different variants of Huffman encoding

ERICSSON

SigComp Overview and extended operation

Hans Hannu
hans.hannu@epl.ericsson.se

ERICSSON

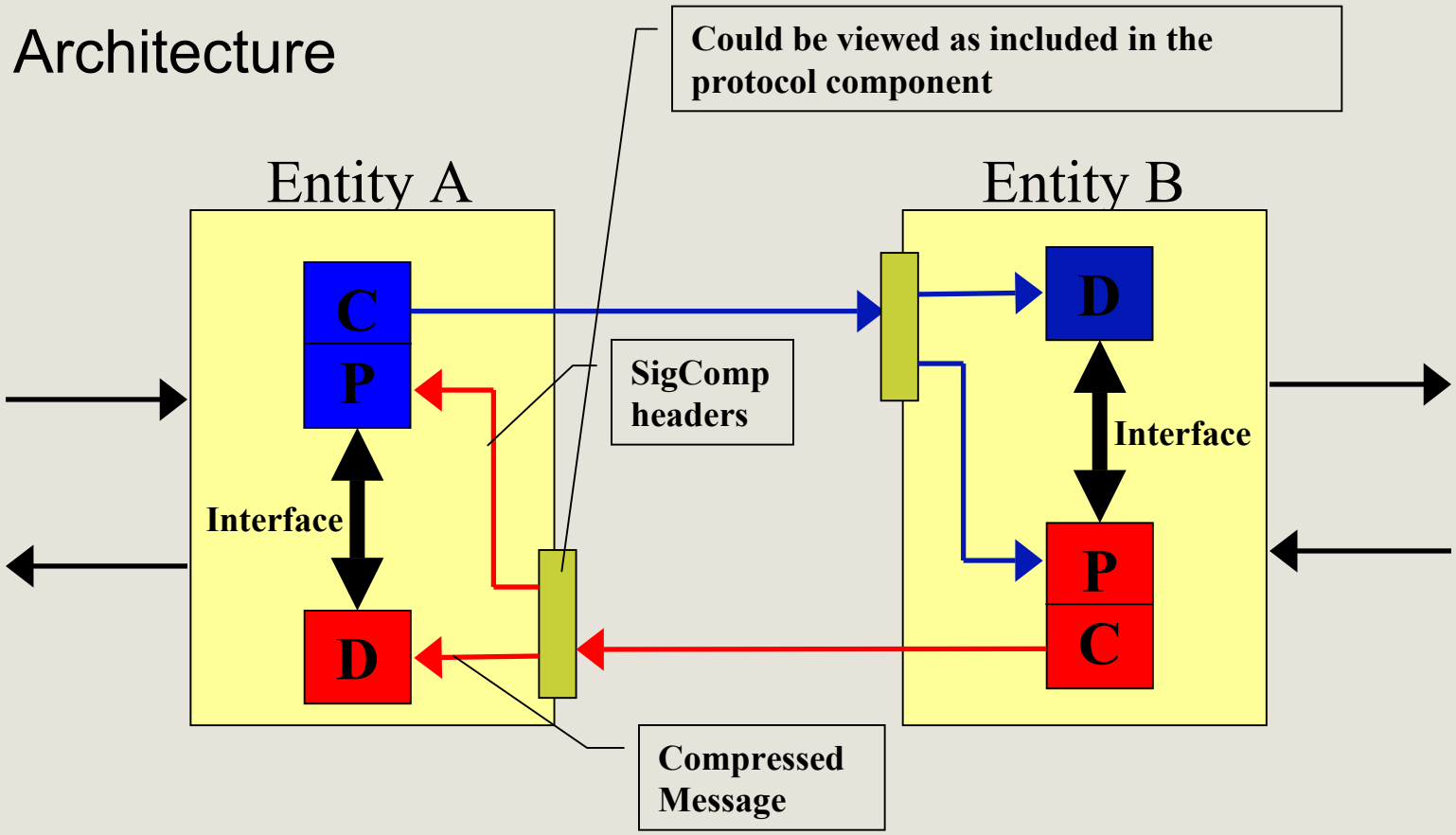
SigComp [*draft-ietf-rohc-sigcomp-02.txt*], 1(4)

- Protocol Component
  - Acknowledgement procedure, etc
  - Improved compression ratio.
- Compression Framework Component
  - “Universal” Decompressor, etc
  - Flexibility
    - Compression algorithms
    - Allows for different compression algorithms in UL and DL

ERICSSON

SigComp [*draft-ietf-rohc-sigcomp-02.txt*], 2(4)

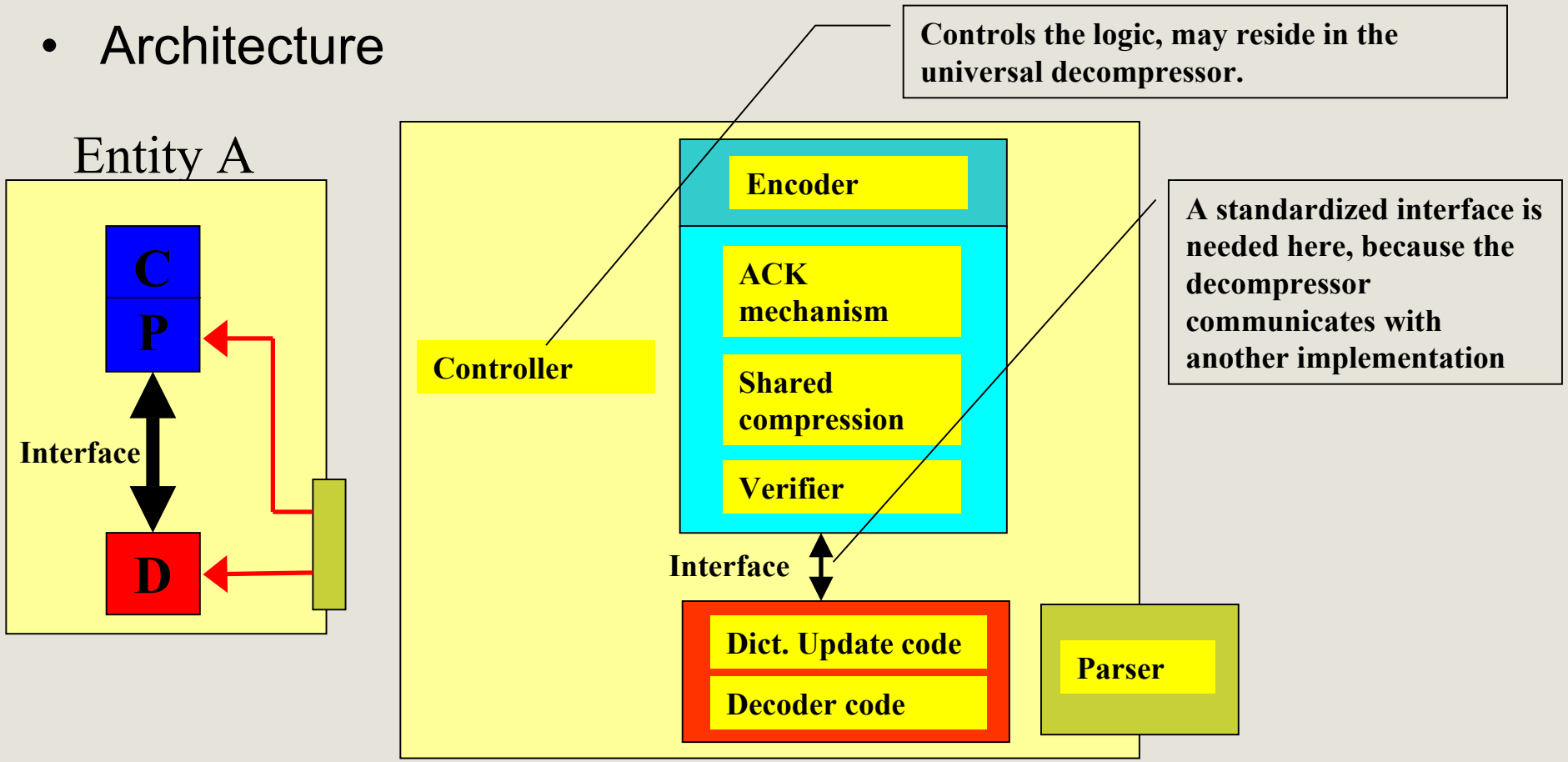
- Architecture



ERICSSON

SigComp [*draft-ietf-rohc-sigcomp-02.txt*], 3(4)

- Architecture



ERICSSON

SigComp [*draft-ietf-rohc-sigcomp-02.txt*], 4(4)

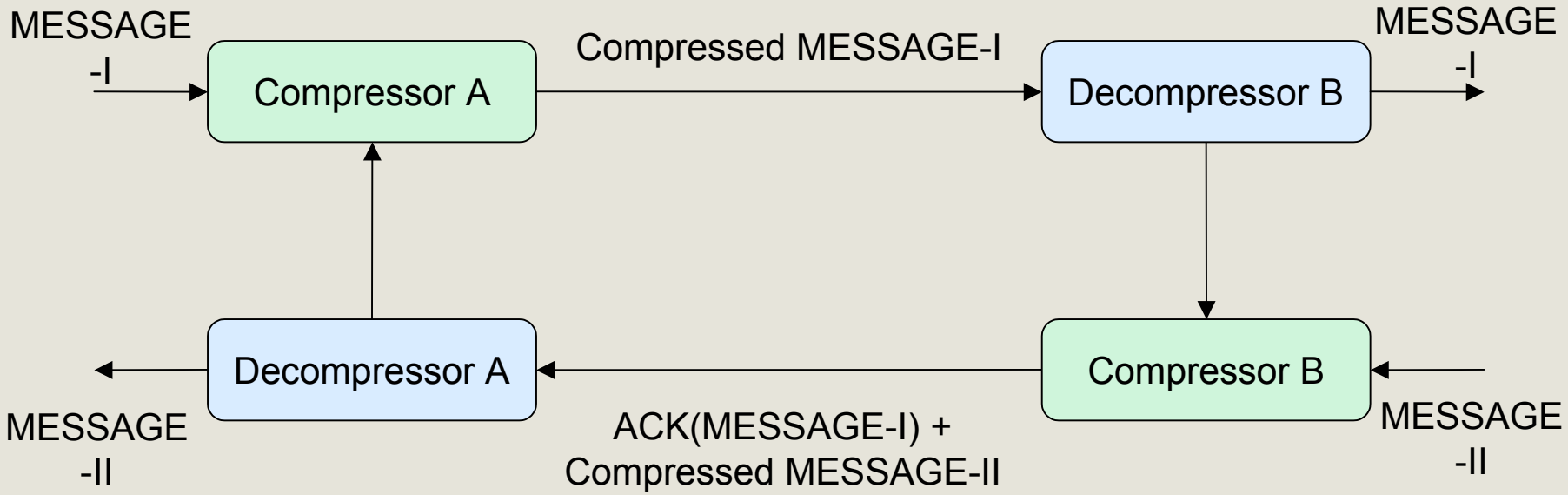
- Modular solution
  - Per-message compression
  - Dynamic compression
  - Shared compression
- Capability exchange
  - 4 Parameters

The authors believe that there might be IPR issues related to the extended operation mechanisms. For more information refer to:

http://www.ietf.org/ipr.html

ERICSSON

Extended operation -
Dynamic compression with Explicit Acks, 1(2)



- Optional - established in the capability exchange
- Robust compression for unreliable transport
  - Dynamic compression

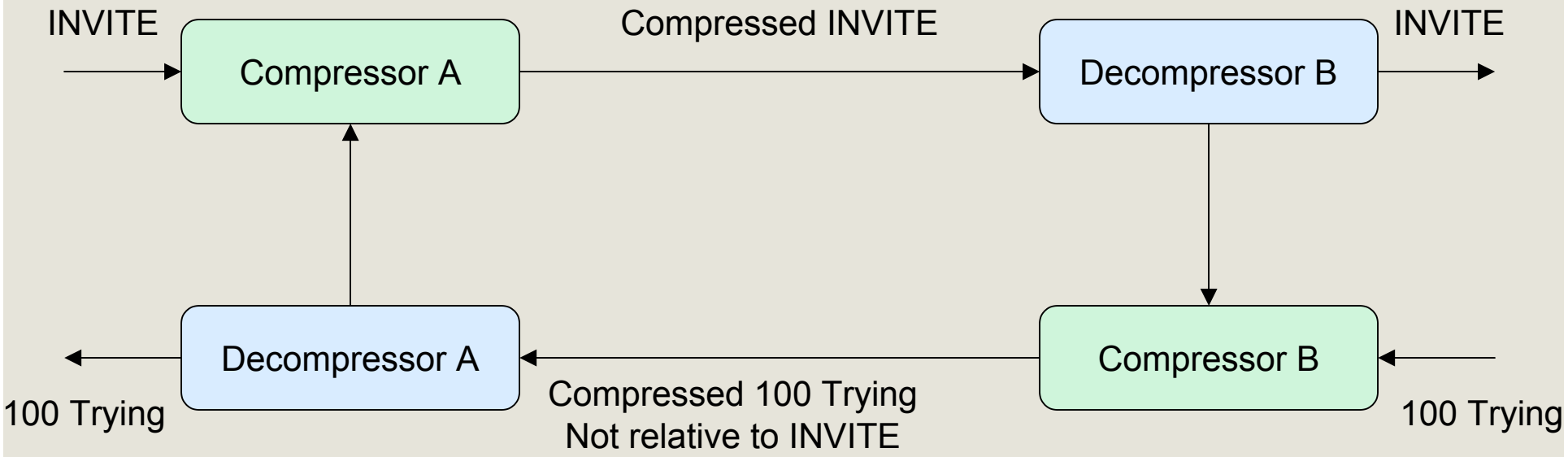
ERICSSON

Extended operation -
Dynamic compression with Explicit Acks, 2(2)

- Dynamic compression in conjunction with Shared compression is made possible
- Header attached to the compressor output
  - MID
  - ACKs
- Three way handshake

ERICSSON

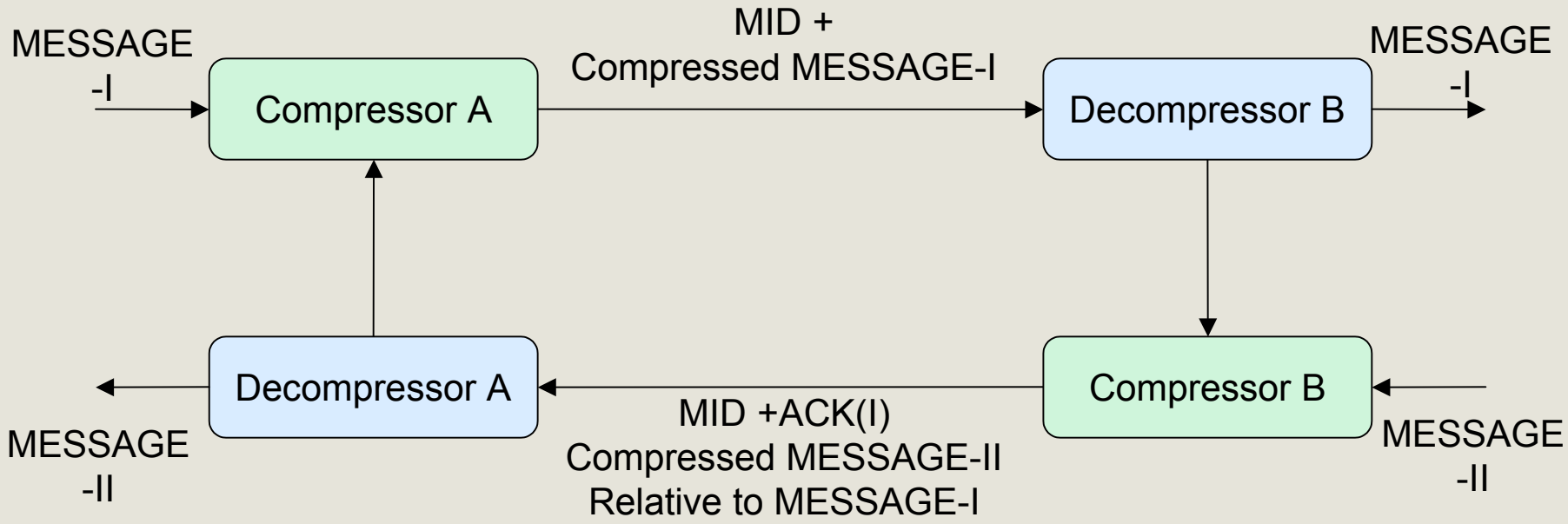
Extended operation -
Dynamic compression with Implicit Acks



- 100 Trying message is sent in response to INVITE
- Compressor infers that INVITE message has arrived
  - Can compress dynamically relative to INVITE for next coming messages

ERICSSON

Extended operation -
Shared compression, 1(3)

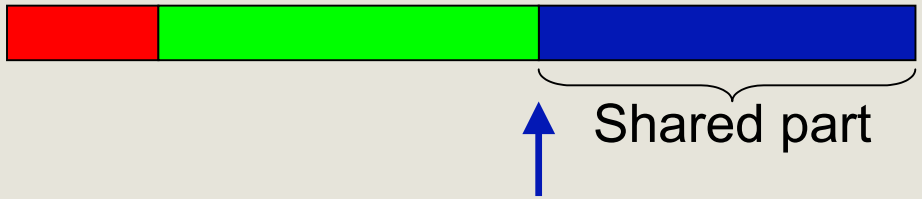


- Optional - Support established in the capability exchange
  - No need to use even if there is support
- Received messages are used in the compression process

ERICSSON

Extended operation -
Shared compression, 2(3)

- shared_start
  - Set by the sending entity's compressor
  - Zero indicates no use of shared compression



- shared_length
  - Set by the receiving entity's protocol
  - Informs the decompressor if new information is written to the shared part of the byte buffer

ERICSSON

Extended operation -
Shared compression, 3(3)

- Increase of compression ratio
- Test
  - Sequence A2 in draft-ietf-rohc-signaling-req-assump-03
    - 14 messages 6563 bytes
  - DEFLATE, DD size 4096, FIFO approach

Transport	Dynamic + Shared Comp.	Dynamic Comp.
Unreliable	993 (~6.6)	1448 (~4.5)
Reliable	988 (~6.6)	1328 (~4.9)

ERICSSON

Open issues, 1(3)

- Explicit acknowledgement scheme
  - Controller, external to the universal decompressor, or
    - Some extra byte buffer entries?
  - A hook in the universal decompressor?
    - Some extra tokens?
  - Is there a difference?
- Shared compression
  - Capability exchange, or
  - Activation internally in SigComp?
    - Byte buffer entry (entries)?
    - Token activated?

ERICSSON

Open issues, 2(3)

- Functionality provided by SigComp headers
  - CID?
  - Decompressor feedback?
  - Parameter “values”?
- Header formats
  - Efficient Standardized set of headers, or
  - Non optimized header format(s)?
    - Compressed together with the actual message
    - Tokens loaded to universal decompressor to understand headers

ERICSSON

Open issues, 3(3)

- Interface between protocol and universal decompressor
  - Dependent on whether the controller is external to the universal decompressor
    - Byte buffer entries, or
    - Tokens?
  - Both approaches require
    - Mapping functionality

Signaling compression: way forward

- How many documents?
  - Requirements -- I
  - Universal decompressor virtual machine -- PS
  - Protocol/Framework -- PS
  - Example UDVM decompressors – I (IPR, later?)
  - Example extended interactions – I (IPR, later???)

Work on them now!

- Assumption: extended schemes work on base protocol
  - Need hooks in base protocol and in UDVM
- If that does not seem to work:
  - Third document: protocol for extended interactions – PS (IPR)
- "Benchmark" info with flow, dictionary, UDVM code

52nd IETF: Agenda (Thu afternoon)

- 1530 TCP
  - 1530 TCP field behavior — West (10)
  - 1540 Requirements document ➡ freeze? — Jonsson (10)
  - 1550 Role of EPIC — West (10)
  - 1600 Progress in merging the drafts — (Authors) (10)
  - 1610 Requirements unmet so far — Jonsson (10)
- 1620 SCTP — Schmidt (20)
- 1640 MIB — Quittek (20)
- 1700 RTP ➡ DS — Chairs (10)
- 1710 Rechartering — (20)

Roke Manor Research

TCP Behaviour

Mark West
mark.a.west@roke.co.uk

The ‘TCP Model’ document

Roke Manor Research

- First attempt at an I-D: draft-west-tcpip-field-behavior-00
- Performs an analysis of TCP field behaviour
- Some comments and typos received
- Any more welcome…

Issues

Roke Manor Research

- Much of the behaviour is straightforward
- However, there are issues arising, including:
  - Checksums
  - Robustness / Efficiency
  - Transparency / Efficiency
  - ECN
  - Bi-directionality
  - Parallel flows
  - Interaction with pilc

Checksums

Roke Manor Research

- TCP checksum will be carried end-to-end
  - It is the only end-to-end validation
- Is the TCP checksum useful to verify decompression?
  - Doesn’t verify all IP fields
  - Simple checksum, so known flaws
  - Needs to be computed over payload as well
- Should an alternate/additional mechanism be used to verify correct decompression?

Robustness / Efficiency

Roke Manor Research

- ROHC RTP is very ‘packet centric’
  - RTP runs over a datagram service
- TCP is a byte-stream
- For example, there is (generally) no stable mapping between packets and sequence number
  - Bulk data transfer comes closest
  - But even then the MSS can vary between flows
- Need to be careful about this…

Transparency / Efficiency

Roke Manor Research

- There are reserved bits in the TCP header
- Sometimes people find a use for them, e.g. ECN
- Proposals already exist for some of the flags that remain (e.g. EIFEL)
- Transparency means that the compressor will not ignore these bits
  - Could fail to compress headers using these bits
  - Could support these bits changing (in currently undefined ways)
- Supporting changing bits is desirable for efficiency and future-proofing
- May need to be careful how to handle these bits…

ECN

Roke Manor Research

- ECN is a particular example of varying behaviours
- There are 2 distinct flavors – original and ECN with nonces
- Very different from a compression perspective
- Also, assumption that ECN is not used on ACKs is challenged in schemes such as ACK Congestion Control

Bi-directionality

Roke Manor Research

- Two distinct deployment scenarios:
  - Separate compression/decompression for each direction
  - Shared compression/decompression
- If we can assume that in some cases a co-located compressor and decompressor can share information, does this offer any benefits?

Bi-directionality

Roke Manor Research

- Examples:
  - Ack number prediction
    Sequence numbers and packet sizes in the forward path can be used to predict acknowledgement numbers
  - Implicit acknowledgements
    TCP acknowledgements can be translated into compressor acknowledgements

Parallel Connections

Roke Manor Research

- May have many TCP connections between the same two hosts
- IP header is largely common
- Would improve efficiency (especially for short-lived connections) to share this state
- Some TCP fields may be ‘close’ to values used for an existing connection
  - Ephemeral port selection
  - Initial Sequence Number selection

Interaction with pilc

Roke Manor Research

- ASYM identifies weaknesses with compression schemes
- ROHC-TCP intends to address
  - Compression of options
  - Packet loss degrading efficiency
  - Support for tunnel encapsulations
- describes many ‘ACK munging’ schemes
  - ACK filtering, decimation and reconstruction can all be done ‘in series’ with compression
  - ACK companding could be supported by ROHC
    Depends in part how the companding data is carried
- Techniques that rely on looking inside the header cannot easily be used *after* compression…

Interaction with pilc

Roke Manor Research

- RFC 3150 [SLOW] and [LINK] discuss header compression
  - And give a nice advert for ROHC ☺
- RFC 3150 also
  - identifies support for option compression
  - contains guidelines for MTU selection – which will directly affect TCP MSS

Conclusion

Roke Manor Research

- The behaviour analysis gives us a starting point for defining TCP compression
- It also gives us some questions and other issues
- Plan:
  - Rev. the draft
  - Take the discussion to the list

TCP Requirements Update

Lars-Erik Jonsson

lars-erik.jonsson@ericsson.com

ROHC@IETF52, SLC

December 2001

TCP Requirements - Brief recapitulation

- Robustness (next slide)
- Efficient compression of ECN bits
- Compress when TCP options, and compress some, e.g.
  - SACK option
  - Timestamp option
- Improved efficiency for short-lived TCP transfers
  - E.g., web accesses with 2-3 data segments + 7 segment overhead
- Availability to the Internet at large
  - Important to avoid encumbered solutions

TCP Requirements - Robustness

- Residual errors in compressed headers
  - Links used for TCP are used to deliver a low residual error rate
  - No need for explicit mechanisms to avoid residual errors to propagate
  - Must not affect TCP's mechanisms for error detection
    - TCP checksum
- Losses between compressor and decompressor
  - Scheme must provide mechanisms to **avoid** losses to
    - propagate in more losses, *or*
    - cause undetected context damage that might result in generation of incorrect subsequent headers
  - Various TCP mechanisms can tolerate and quickly recover from some packet loss. Header compression should not disable (might instead help) such mechanisms

TCP Requirements - Open issues

- Compression in tunnels means possible misordering between compressor and decompressor
  - Should this be
    - Prohibited?
    - Allowed with requirement on detection?
    - Generally allowed?
  - Framework issues, not only for TCP profile

TCP Requirements - What now?

- Final update?
- Freeze call in ROHC, TsvWG, PILC

Roke Manor Research

The role of EPIC(-lite)

Mark West
mark.a.west@roke.co.uk

EPIC / EPIC-lite

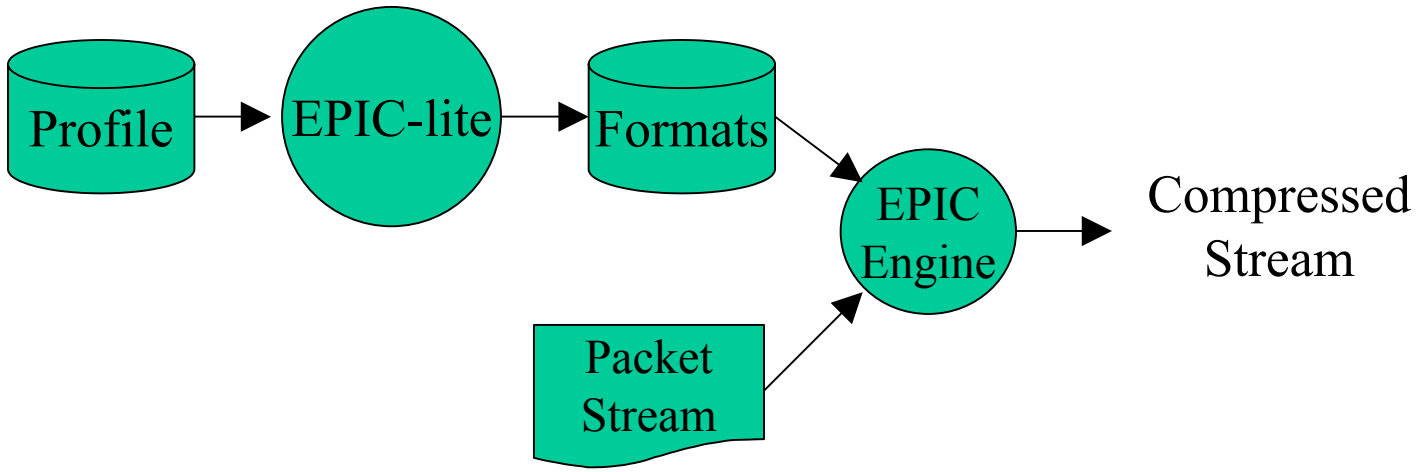
Roke Manor Research

- EPIC and EPIC-lite specifically refer to algorithms
  - EPIC-lite is simple and efficient
  - EPIC is optimally efficient (and is encumbered with IPR claims)
- EPIC Framework is used generally to refer to the common framework used by this pair of algorithms

What EPIC is…

Roke Manor Research

- EPIC is about generating packet formats
  - Allows the packets between compressor and decompressor to be described at a higher level
  - Automatically generates highly efficient formats
- The description can be used to compress and decompress headers in a generic way



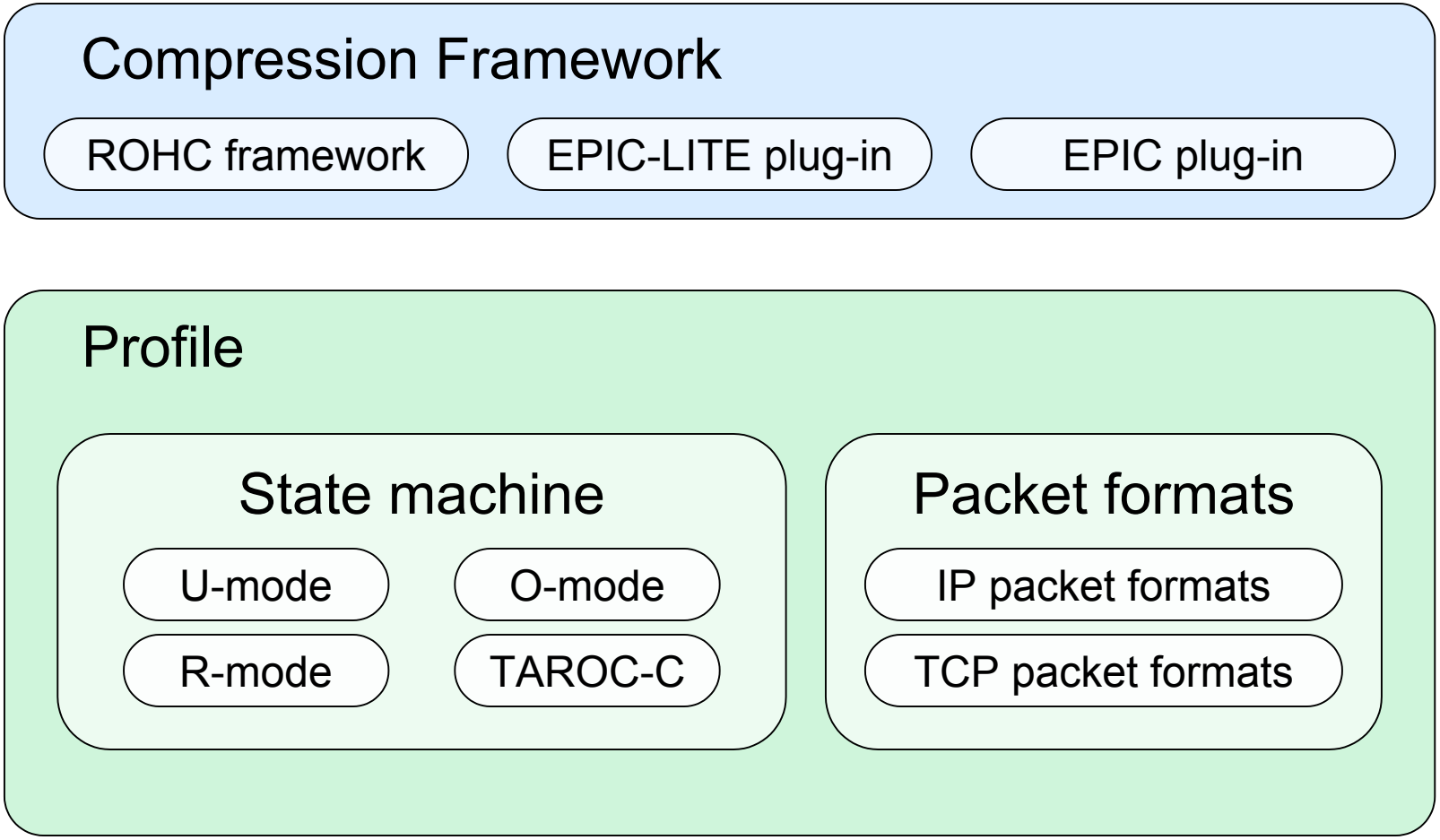
What EPIC is not…

Roke Manor Research

- It is not a complete framework for header compression
  - EPIC-lite needs something like the ROHC framework (established for RTP) to drive it

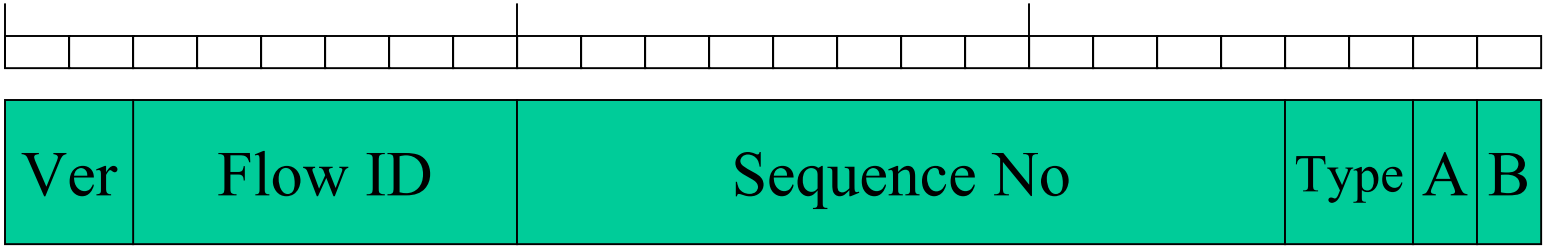
Architecture

Roke Manor Research



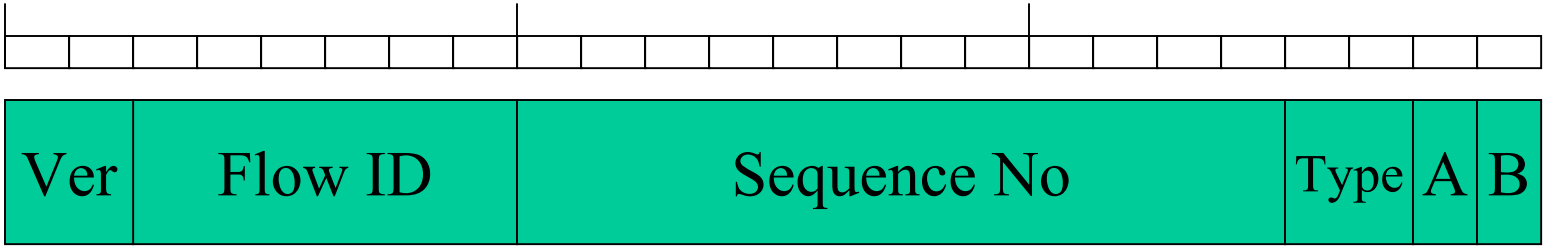
An example

Roke Manor Research



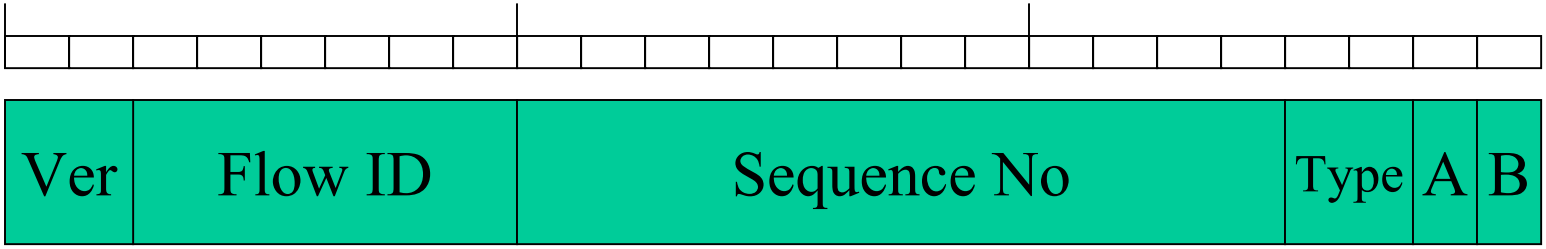
What would a profile look like for this simple protocol header?

An example



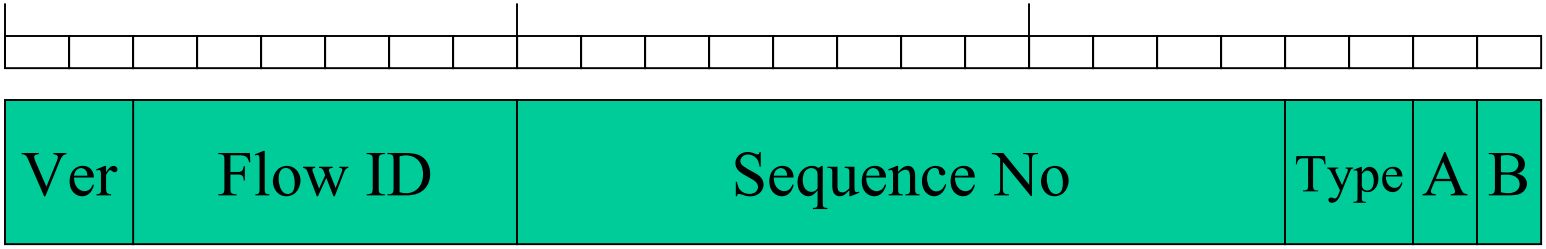

```
Version = STATIC-KNOWN(2,1)
```

An example



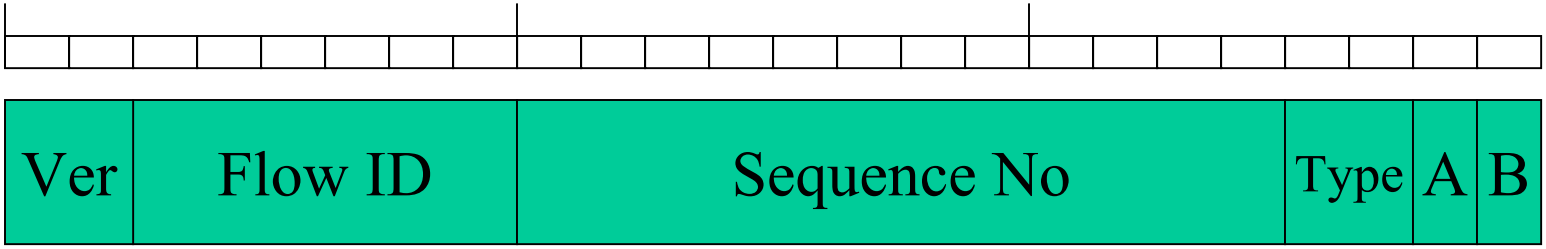

```
Flow-ID = STATIC-UNKNOWN(6)
```

An example



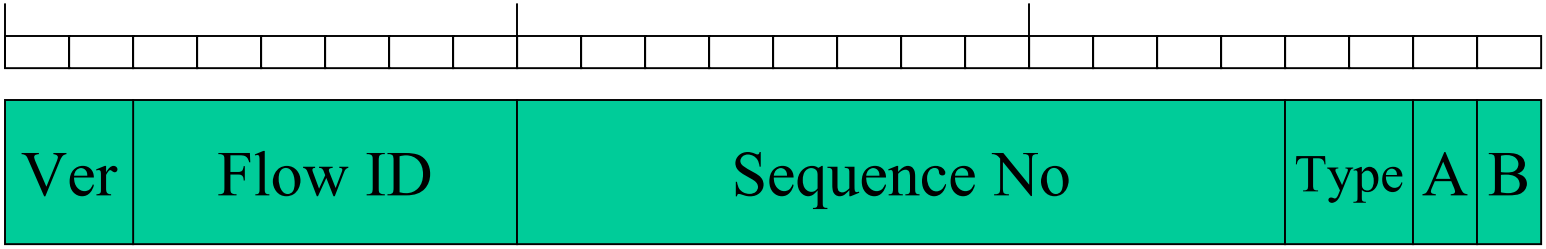

```
Sequence-number =
      LSB(4,-1,90%)       |
      LSB(8,-1, 5%)       |
      IRREGULAR(12, 5%)
```

An example



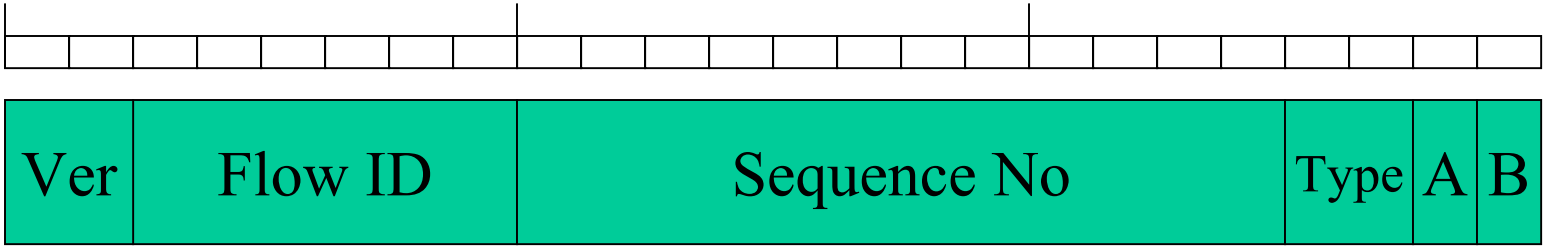

```
Type =
      STATIC(90%)          |
      IRREGULAR(2,10%)
```

An example




```
Flag-A = IRREGULAR(1,100%)
```

An example




```
Flag-B =
      VALUE(1,0,80%)   |
      VALUE(1,1,20%)
```

Example Packet Formats

Roke Manor Research

- The description gives 12 packet formats
  - 3 choices for encoding the counter
  - 2 choices for the ‘type’ field
  - 2 explicit values for the second flag
- EPIC-lite builds the compressed packet formats and assigns each one a Huffman code as a prefix
- The prefix identifies precisely *which* encoding was chosen for each field

EPIC-lite generated formats

Roke Manor Research

INDICATOR	'A'	COUNTER	TYPE	('B')	%
0	X	XXXX		(0)	64.8
10	X	XXXX		(1)	16.2
110	X	XXXX	XX	(0)	7.2
11100	X	XXXXXXXXXXXX		(0)	3.6
11101	X	XXXXXXXX		(0)	3.6
11110	X	XXXX	XX	(1)	1.8
1111100	X	XXXXXXXXXXXX		(1)	1.8
1111101	X	XXXXXXXX		(1)	0.9
1111110	X	XXXXXXXXXXXX	XX	(0)	0.9
11111110	X	XXXXXXXX	XX	(0)	0.4
111111110	X	XXXXXXXXXXXX	XX	(1)	0.4
111111111	X	XXXXXXXX	XX	(1)	0.1

Example compressed packet

Roke Manor Research

- So, if the compressor selected
  - 4 LSBs for the counter
  - The value of the type field (IRREGULAR)
  - IRREGULAR for Flag-A (as the only choice)
  - Value ‘1’ for Flag-B
- The compressed packet format would be:

  `11110 X XXXX XX`
- Note the difference between this approach and ROHC-RTP
  - EPIC assumes that the encoding choices are made per-field
  - ROHC-RTP extracts the field changes and then selects the ‘best match’ header

Is that it?

Roke Manor Research

- After the EPIC-lite generation algorithm has been run, the packet formats are created
  - EPIC does essentially the same, but applies Huffman encoding to the entire compressed header
- It is theoretically possible to simply take the packet formats and write a compressor / decompressor for the protocol based on these
- Note that because EPIC treats fields independently, many formats can be created
- This is beneficial because the compressed formats closely match the described protocol behaviour
- The formats also rely upon the compressor and decompressor having the same definition of each of the encoding methods
  - This is implicitly true of ROHC-RTP

Packet Compression

Roke Manor Research

- The EPIC(-lite) framework also describes how to use the profile to compress a header
- The description is not exhaustive (there are local implementation choices)
- It also needs an external mechanism to handle, for example, feedback
- But there is enough information in the profile to compress a header…

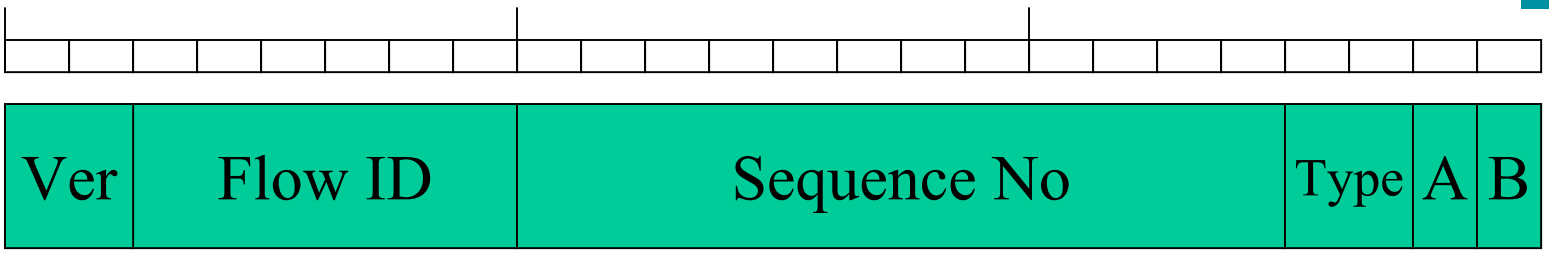
Compressing a packet

Roke Manor Research

- The compressor works through each field in turn
- For each field it has to select an encoding
- If multiple encodings could be used, the choice is left to the compressor
- Each encoding tells the compressor how large the field is
- Choosing an applicable coding consumes bits from the uncompressed header and may add bits to the compressed header
  - More complex encodings may also generate new bits to be compressed
  - This is equivalent to the NBO flag in RFC 3095, for example
  - EPIC just treats these as new fields to compress
- When encoding is complete the set of selected codings maps to a packet format

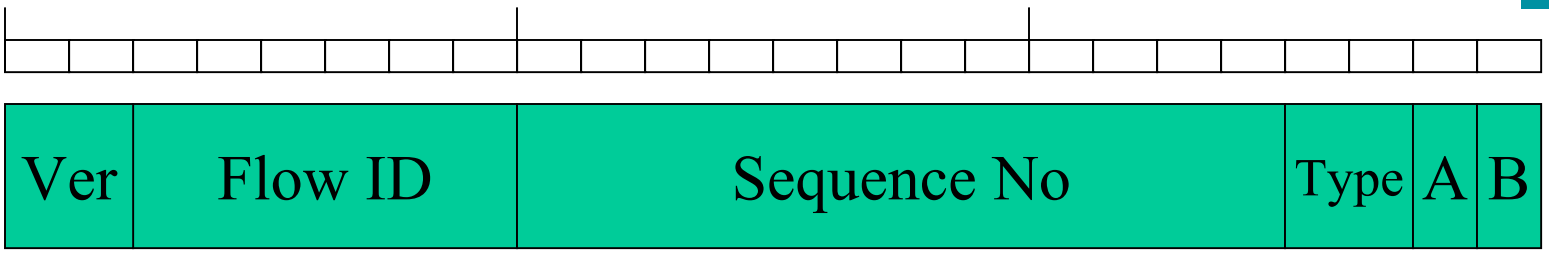
Compression Example

Roke Manor Research



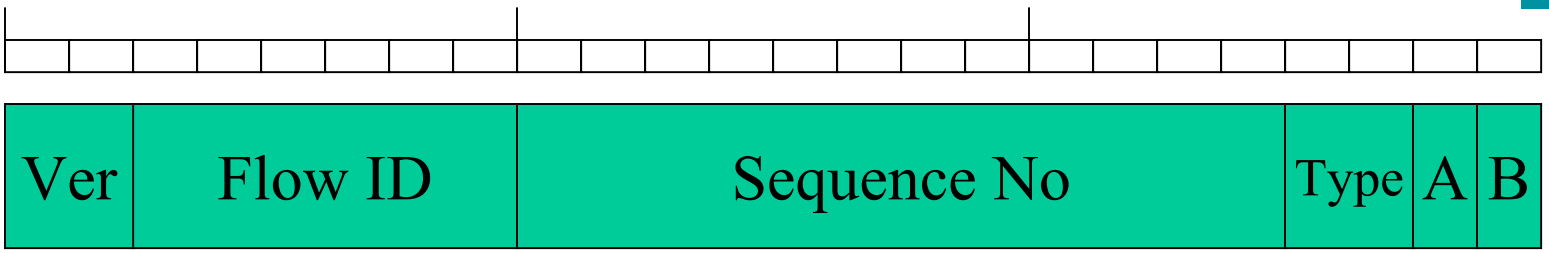
- Compressing the Version simply consumes 2 bits and checks that they have the correct value
- The Flow-ID
  - Is sent and set in IR packet
  - Subsequently the bits are simply consumed and checked

Compression Example



- Sequence number is compressed exactly as for WLSB described in RFC 3095
  - 12 bits are extracted from the uncompressed header
  - Each LSB encoding is checked against this value and the context
  - A selection is made by the compressor of any of the encodings that fit
  - The IRREGULAR encoding is a ‘catch-all’

Compression Example



- The Type field can only be STATIC if all entries in the context history are the same and match the current value
  - This is the same as not transmitting a value in RFC 3095
- Flag A is always carried, so 1 bit is moved from the uncompressed to the compressed data
- Flag B makes an exact match on the value
  - This choice influences the indicator

Decompression

Roke Manor Research

- The decompressor matches the indicator
  - Use of Huffman codes makes this easy
- The indicator can be mapped to the precise definition of which encodings were used by the compressor
- A similar process to compression is used to reconstruct the uncompressed header
- Without giving an explicit example…
  … read through the previous slides backwards!

In reality

Roke Manor Research

- There are more encoding methods than shown in the example!
- Some of these are relatively sophisticated
- But fundamentally work in the same way
- They are designed to allow accurate descriptions of more complex protocols (such as RTP, TCP, SCTP, …)

Why use EPIC-lite?

Roke Manor Research

- Allows high-level description of protocol behaviour
  - Easy to work with
  - Facilitates re-use of descriptions of protocol layers
- One-time cost for implementing EPIC-lite framework
  - Second and subsequent protocols are free ☺
- Using EPIC-lite to do compression and decompression allows use of large number of packet formats
  - Compressed formats more closely match behaviour increasing overall compression efficiency

Microsoft Research Asia

TAROC-C – the Control Mechanism of TAROC (TCP-Aware RObust header Compression scheme)

http://www.ietf.org/internet-drafts/draft-ietf-rohc-tcp-taroc-04.txt
http://www.ietf.org/internet-drafts/draft-ietf-rohc-tcp-epic-02.txt

Qian Zhang
Microsoft Research

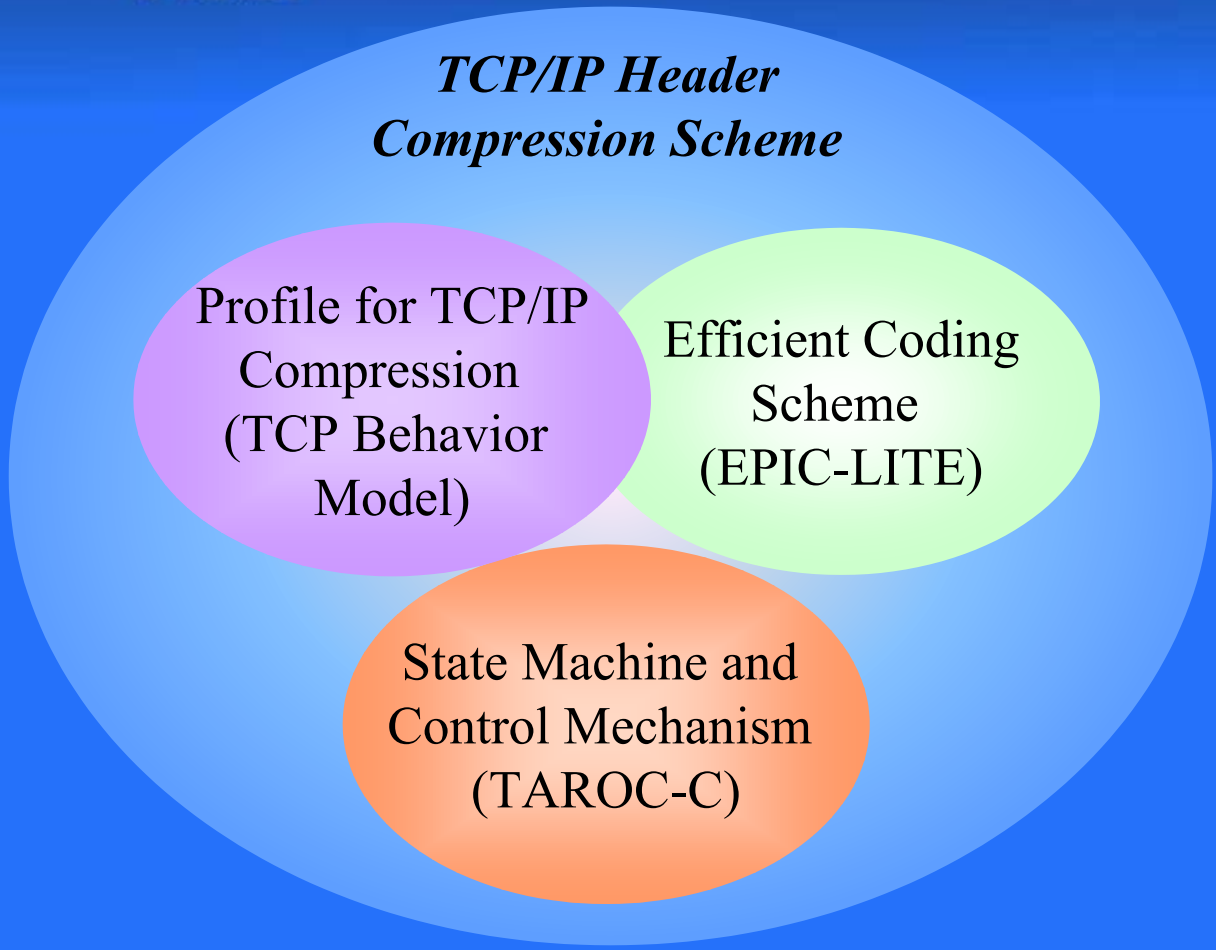
Outline

Microsoft Research Asia

- Key Components for TCP/IP Header Compression Scheme
- Key Concepts of TAROC-C
  - ✓ TCP congestion window tracking algorithms
  - ✓ State machine of TCP/IP header compression scheme
  - ✓ No IPR statement for TAROC-C
- Some Open Issues for State Machine
  - ✓ Acknowledgement path optimization
  - ✓ Context sharing

Key Components for TCP/IP Header Compression Scheme

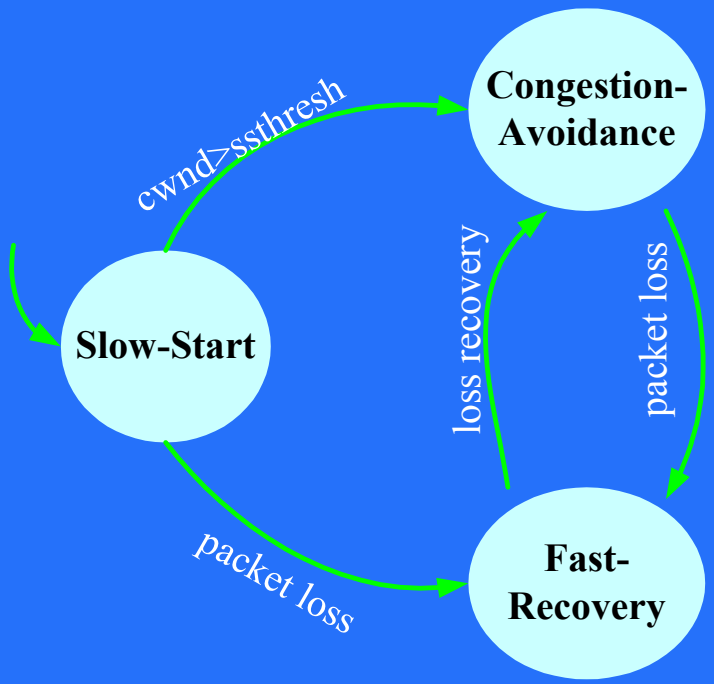
Microsoft Research Asia



Congestion Window Tracking Algorithms Modification

Microsoft Research Asia

- Robustness of TAROC-C
  - ✓ Window-based LSB encoding
- Efficiency of TAROC-C
  - ✓ Tracking-based TCP congestion window estimation
- Improvement
  - ✓ Clarify initialization process
    - the first segment is not necessary to be the SYN segment
  - ✓ MIN/MAX boundary of estimated TCP congestion window

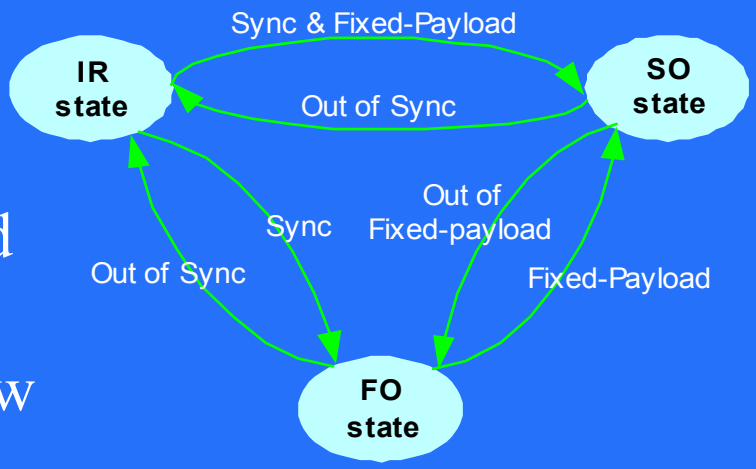


State Machine Modification

Microsoft Research Asia

- FO/SO state → IR state
  - ✓ When 1: VSW contains only one packet, which means there is a long jump in the packet sequence number or acknowledgement number

    |seqno (ackno) - CMAXSN (CMAXACK)| > estimated congestion window size
  - ✓ When 2: static context of transfers changed
  - ✓ Action: transit to the IR state and re-initialize the algorithm for tracking TCP congestion window



Open Issues for State Machine (1)

Microsoft Research Asia

- Acknowledgement Path Optimization
  - ✓ Pros
    - Significant spectral efficiency in the acknowledgement direction
  - ✓ Cons
    - Increase burst size at the TCP sender side
    - Fewer ACKs slow down the rate of growth of the cwnd
    - Fast Retransmission and Fast Recovery algorithms are less effective when ACKs are lost
  - ✓ Issues to be addressed:
    - How to maintain the ACK-clock
    - How to maintain the evolution of TCP's cwnd
    - How to reduce the burst

Open Issues for State Machine (2)

Microsoft Research Asia

- Context Sharing
  - ✓ Fields that can be shared
    - Source address
    - Destination address
  - ✓ Fields that may be shared
    - IP-ID
    - Port
    - Sequence number
    - Depends on the concrete implementation

Preliminary Profile for TCP Options

Microsoft Research Asia

- **method TCP-WINDOW-SCALE**

  encode Kind as STATIC-KNOWN(8,3)
  encode WS_Length as STATIC-KNOWN(8,3)
  encode WS_Count as IRREGULAR(8)
  end_method

- **method TCP-TIMESTAMP**

  encode Kind as STATIC-KNOWN(8,8)
  encode TS_Length as STATIC-KNOWN(8,10)
  encode TS_Value as LSB(8,-1) 80% or LSB(16,-1) 19%
  or LSB(24,-1) 0.9% or IRREGULAR(32) 0.1%
  encode TS_Echo_Reply as LSB(8,-1) 90% or LSB(16,-1) 19%
  or LSB(24,-1) 0.9% or IRREGULAR(32) 0.1%
  end_method

Preliminary Profile for TCP Options

Microsoft Research Asia

- **method TCP-SACK**
  encode Kind as STATIC-KNOWN(8,5)
  encode SACK_Length as INFERRED(8)
  encode Edge as LIST(8,1,8,0, BLOCK, BLOCK, BLOCK, BLOCK)
  encode Edge.Order as VALUE(5,0) 100%
  end_method
- **method BLOCK**
  encode SACK_Block as VALUE(1,0) 50% or BLOCK-PRESENT 50%
  end_method
- **method BLOCK-PRESENT**
  encode Present as VALUE(1,1) 100%
  encode Left_Edge as INFERRED(32)
  encode Right_Edge as INFERRED-OFFSET(32,1)
  encode Base as LSB-PADDED(32,8) 80%
  or LSB-PADDED(32,20) 19.9% or LSB-PADDED(32,32) 0.1%
  encode Right_Edge.Offset as LSB-PADDED(32,8) 90%
  or LSB-PADDED(32,20) 9.9% or LSB-PADDED(32,32) 0.1%
  end_method

Conclusions

Microsoft Research Asia

- Key Concepts of TAROC-C
  - ✓ TCP congestion window tracking algorithms
  - ✓ State machine of TCP/IP header compression scheme
  - ✓ No IPR statement for TAROC-C
- Some Open Issues for State Machine
  - ✓ Acknowledgement path optimization
  - ✓ Context sharing
- TCP Profile + Encoding Scheme + State Machine

→ Efficient TCP/IP Header Compression Scheme

Requirements not met by current proposals

- Improved compression for short-lived TCP transfers
- Compression of options (SACK, Timestamp)
- Tunneling and extension headers
- Robustness / Efficiency / Reordering

TCP issues

- Evaluation of “mode needs” for TCP
  - TAROC can be viewed as a U/O-mode implementation optimization
  - Modes needed in 3095 because minimal headers had different formats in U/O and R modes. Current proposals use same formats in all modes
  - Is the mode distinction needed?
- Decompression verification mechanism
  - TCP checksum sufficient?

52nd IETF: Agenda (Thu afternoon)

- 1530 TCP
  - 1530 TCP field behavior — West (10)
  - 1540 Requirements document ➡ freeze? — Jonsson (10)
  - 1550 Role of EPIC — West (10)
  - 1600 Progress in merging the drafts — (Authors) (10)
  - 1610 Requirements unmet so far — Jonsson (10)
- 1620 SCTP — Schmidt (20)
- 1640 MIB — Quittek (20)
- 1700 RTP ➡ DS — Chairs (10)
- 1710 Rechartering — (20)

Requirements for SCTP compression

(Stream Control Transmission Protocol)

Christian Schmidt

52. IETF / RoHC in Salt Lake City

13.12.2001

Christian.Schmidt@icn.siemens.de

Agenda

- Motivation for SCTP compression
- High-level information TCP / SCTP / UDP
- Requirements for SCTP compression
  draft-schmidt-rohc-sctp-requirements-00.txt
- Next?

Christian.Schmidt@icn.siemens.de

Motivation for SCTP compression

- SCTP will be important on mobile access area
  - SCTP as a Transport for SIP (draft-rosenberg-sip-sctp-01.txt)
  - SDP to specify media transport using SCTP (draft-fairlie-mmusic-sdp-sctp-00.txt)
  - SCTP is designed as General Purpose Transport Protocol. The usage of SCTP will be decided by the market.
- Uncompressed SCTP transport is not efficient
- According to the requirements, SCTP compression can be seen as TCP type compression with minor changes.

Christian.Schmidt@icn.siemens.de

High-level information TCP / SCTP / UDP

TCP:

- Connection establishment
- End-to-End flow control with packet retransmission
- Stream oriented

SCTP:

- Connection establishment
- End-to-End flow control with packet retransmission
- Message oriented
- **Multi-streaming**
- **Multi-homing**
- **Dynamic Reconfiguration of IP addresses**

UDP:

- No flow control, no packet retransmission
- No blocking due to lost packets

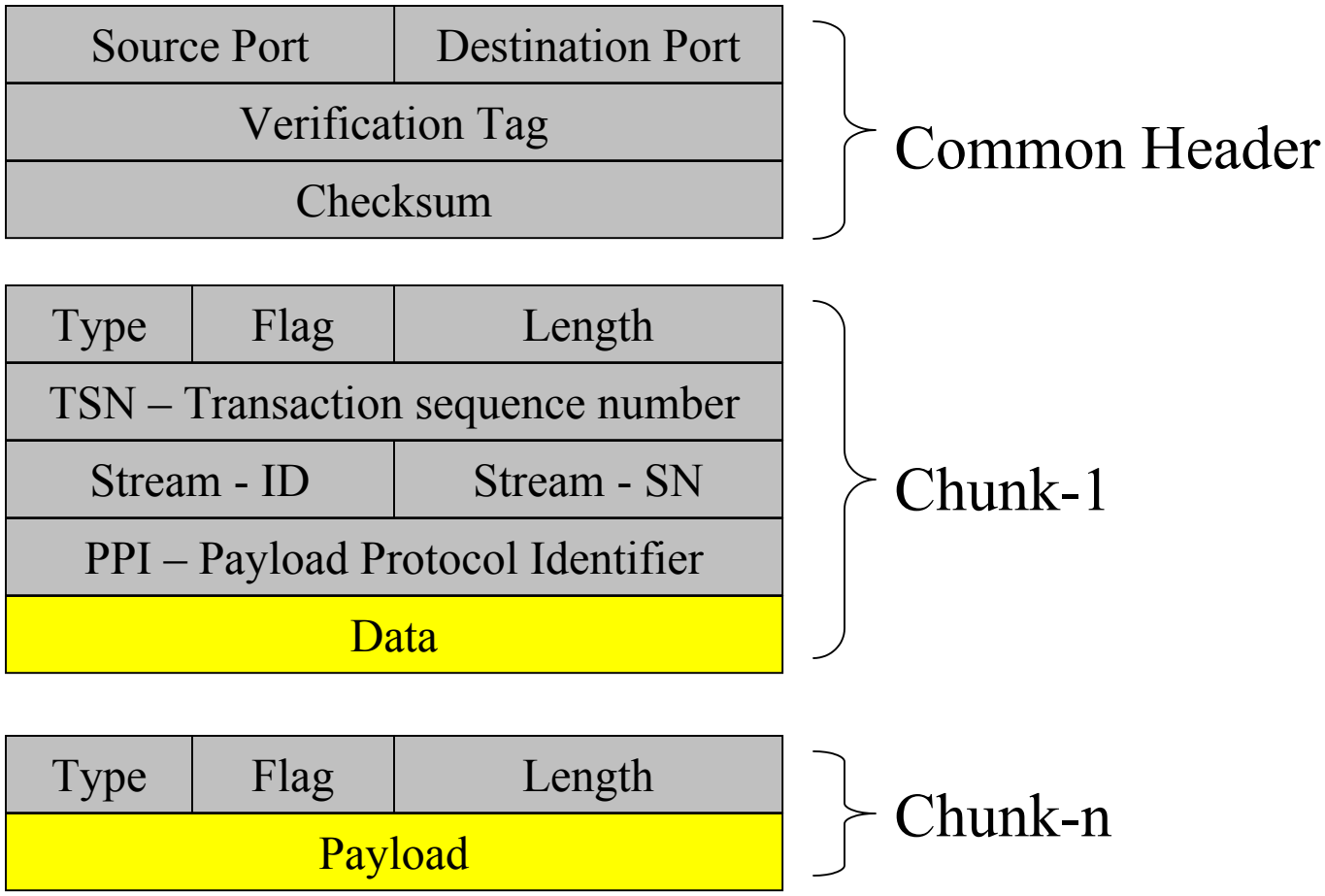
Christian.Schmidt@icn.siemens.de

Requirements for SCTP compression

- Requirements, equivalent with TCP compression (for example efficiency), see Requirements for ROHC IP/TCP Header Compression from Lars-Erik Jonsson
- SCTP specific requirements for compression
  1. SCTP specific protocol structure
  2. SCTP multi-streaming
  3. SCTP extensions

Christian.Schmidt@icn.siemens.de

Req1: SCTP specific protocol structure



Christian.Schmidt@icn.siemens.de

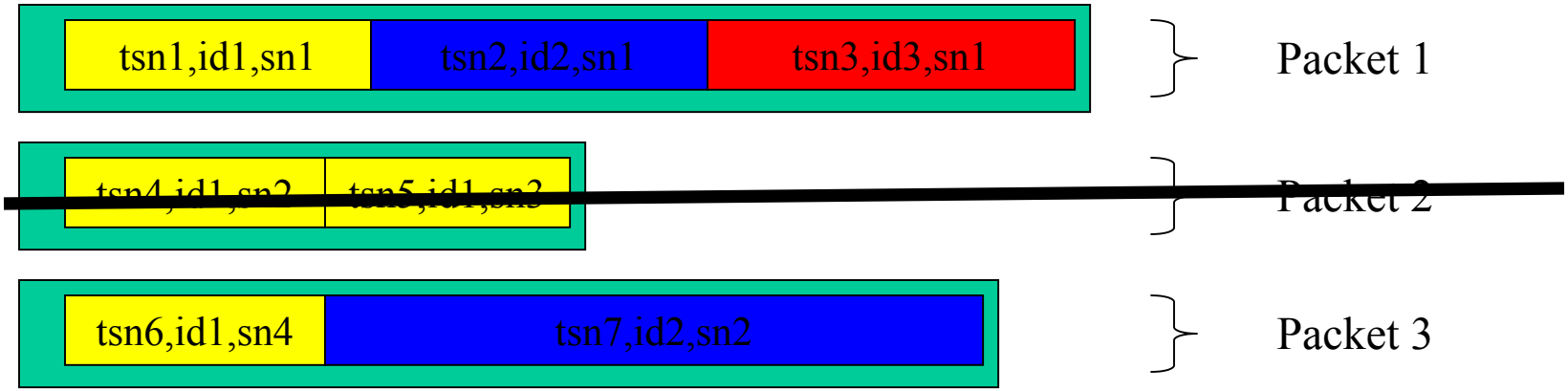
Req2: SCTP multi-streaming

Multi-streaming:

- Partition into multiple streams
- Independent delivery of packets for various streams

Advantages:

- Streams are independent
- Packet loss / damage only has influence to involved streams



Christian.Schmidt@icn.siemens.de

Req2: SCTP multi-streaming

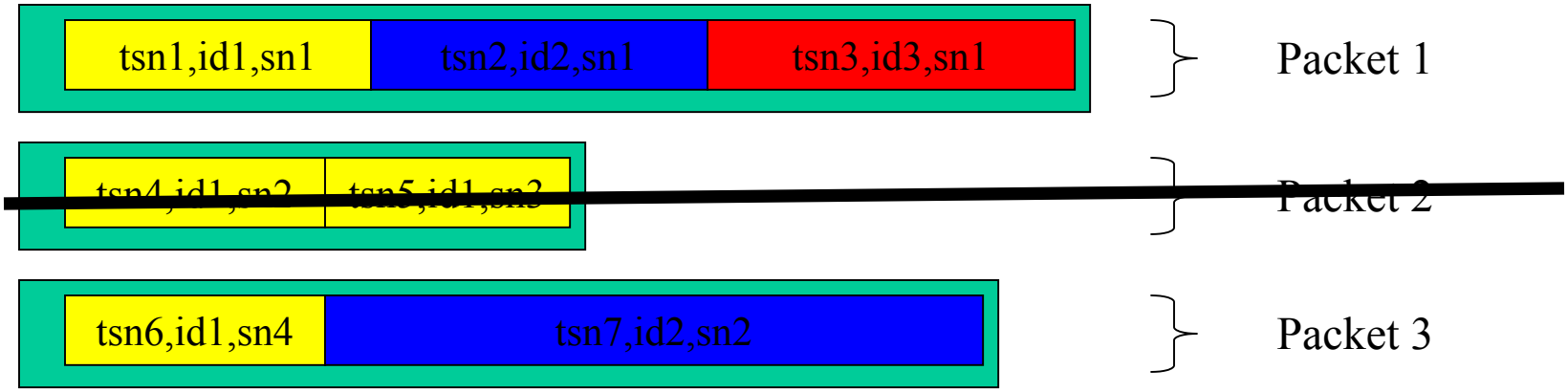
Requirement for SCTP compression:

- Keep SCTP multi-streaming quality of SCTP
- Decompression errors affecting a stream may not influence other streams.

Case 1: Decompression of Packet 3 went well

Case 2: Decompression of Packet 3 fails

Open Issue: How to avoid delay of chunk 7 in this case?



Christian.Schmidt@icn.siemens.de

Req3: SCTP extensions

- SCTP extensions as described for example [ADDIP] should be compressed efficiently. This should also cover new defined chunks.

- Justification:
  SCTP extensions will be a normal part of the protocol. To reach good efficiency for SCTP, these extension have to be handled in an appropriate way.

Christian.Schmidt@icn.siemens.de

Next?

- Acceptance of SCTP compression as RoHC WG Item
- Reissue SCTP Compression Requirement Specification as WG draft
- Further discussions on the mailing list to progress to WG last call.
- Provision of a proposal for SCTP compression as WG draft.

Christian.Schmidt@icn.siemens.de

52nd IETF: Agenda (Thu afternoon)

- 1530 TCP
  - 1530 TCP field behavior — West (10)
  - 1540 Requirements document ➡ freeze? — Jonsson (10)
  - 1550 Role of EPIC — West (10)
  - 1600 Progress in merging the drafts — (Authors) (10)
  - 1610 Requirements unmet so far — Jonsson (10)
- 1620 SCTP — Schmidt (20)
- 1640 MIB — Quittek (20)
- 1700 RTP ➡ DS — Chairs (10)
- 1710 Rechartering — (20)

ROHC-MIB-RTP

<draft-ietf-rohc-mib-rtp-00.txt>

Juergen Quittek <quittek@ccrle.nec.de>
Hannes Hartenstein <hartenst@ccrle.nec.de>
Martin Stiemerling <stiemerling@ccrle.nec.de>
NEC Europe Ltd.

Overview

- Objectives
- MIB Structure
- Object Groups
  - Interface, Header, Channel,
  - Compressor, Decompressor, Statistics
- Discussion Points
  - Channel Ids
  - Compressor Context Issues
  - Statistics
  - Conformance

Objectives

- Monitor running ROHC systems
  - configuration check
  - performance monitoring
  - fault detection
  - fault analysis
- Configure running ROHC systems?
- Context Re-initialization?

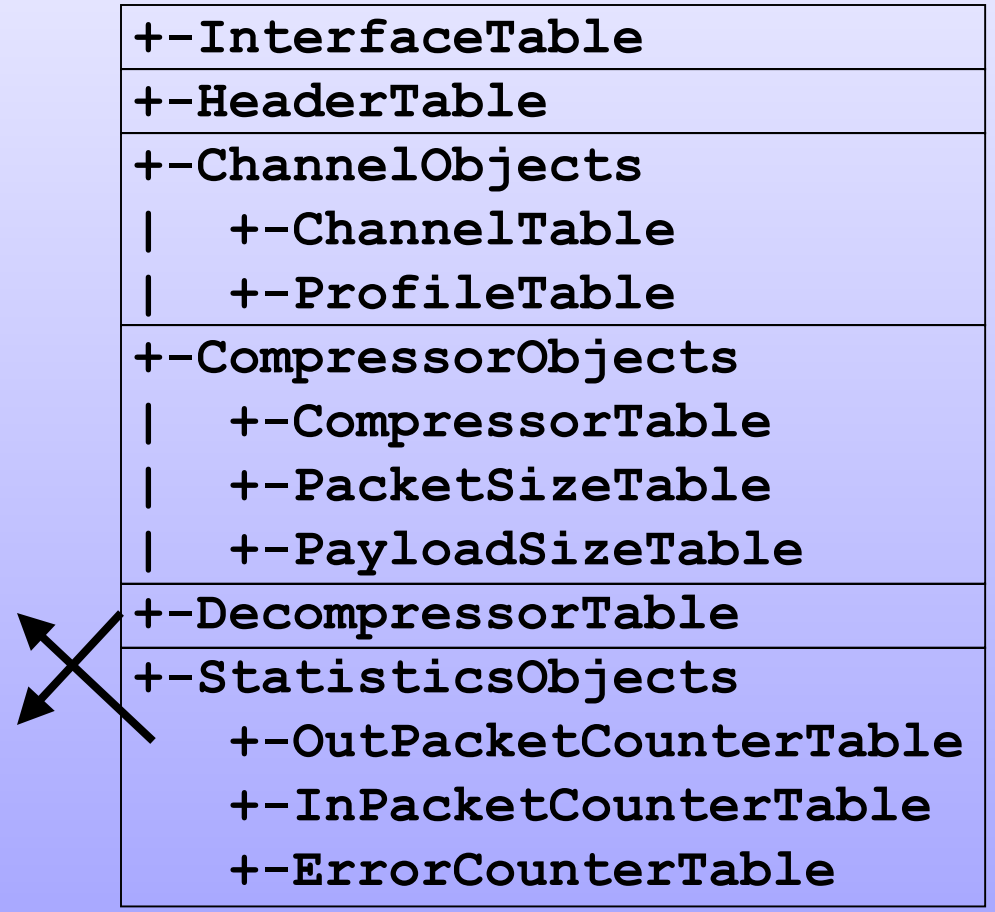
Are there more?

MIB Structure

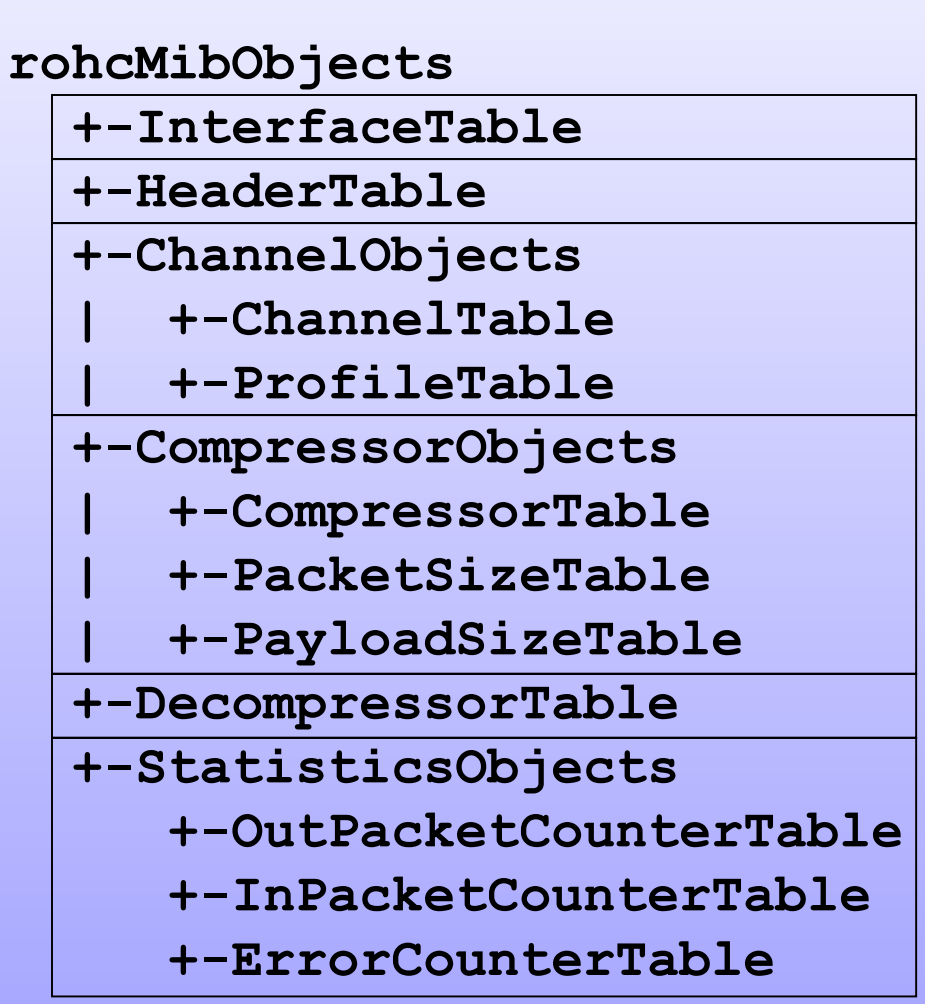
Logical

```
-ROHC-MIB-RTP
  +-Interfaces
     +-Headers
     |
     +-Channels
        +-Profiles
        |
        +-CompressorContexts
        |   +-PacketSizes
        |   +-PayloadSizes
        |   +-OutPacketCounters
        |
        +-DecompressorContexts
           +-InPacketCounters
           +-ErrorCounters
```

Object Tree



MIB Groups



MIB structured in 6 groups

- interfaces group
- header group
- channel group
- compressor group
- decompressor group
- statistics group

Interface Group (rohcIfGroup)

Interfaces implementing ROHC

```
rohcIfTable
  i ifIndex            Integer32 (1..2147483647)
    rohcIfVendor       OBJECT IDENTIFIER
    rohcIfVersion      SnmpAdminString
    rohcIfDescr        SnmpAdminString
    rohcIfClockRes     TimeInterval
    rohcIfStatus       INTEGER {enabled,disabled}
```

Header Group
(rohcHeaderGroup)

Supported header types per interface

```
rohcHeaderTable
  i ifIndex              Integer32 (1..2147483647)
  i rohcHeaderIndex      Integer32,
    rohcHeaderString     SnmpAdminString,
    rohcHeaderDescr      SnmpAdminString
```

Channel Group (1)
(rohcChannelGroup)

Channels per interface

```
rohcChannelTable
  i ifIndex                    Integer32 (1..2147483647)
  i rohcChannelIndex                     RohcChannelIndex
    rohcChannelMaxCID                    Integer32
    rohcChannelLargeCIDs                 TruthValue
    rohcChannelFeedbackFor               RohcChannelIndex
    rohcChannelMRRU                      Integer32
    rohcChannelCompressedFlows           Counter32
    rohcChannelDecompressedFlows         Counter32
```

Channel Group (2)
(rohcChannelGroup)

List of profiles to be used per channel and interface

```
rohcProfileTable
   i ifIndex                Integer32 (1..2147483647)
   i rohcChannelIndex       RohcChannelIndex
   i rohcProfile            Integer32
```

Compressor Group (1)

Compressor contexts per channel and interface

```
rohcCompressorTable
  i ifIndex                      Integer32(1..2147483647)
  i rohcChannelIndex             RohcChannelIndex
  i rohcCompressorCID            Integer32
    rohcCompressorState          INTEGER {ir,fo,so}
    rohcCompressorMode           INTEGER {u,o,r}
    rohcCompressorProfile        Integer32
    rohcCompressorReinit         TruthValue
    rohcCompressorSizesAllowed   Integer32
    rohcCompressorSizesUsed      Integer32
    rohcCompressorTotalRatio     Integer32
    rohcCompressorCurrentRatio   Integer32
    rohcCompressorOutPackets     Counter32
    rohcCompressorInACKs         Counter32
    rohcCompressorInNACKs        Counter32
    rohcCompressorInSNACKs       Counter32
```

Compressor Group (2)
(rohcCompressorGroup)

Allowed and used packet sizes

per compressor context, channel, and interface

```
rohcPacketSizeTable
  i ifIndex                Integer32(1..2147483647)
  i rohcChannelIndex       RohcChannelIndex
  i rohcCompressorCID      Integer32
  i rohcPacketSize         Integer32
    rohcPacketSizeUsed     TruthValue
```

Compressor Group (3)
(rohcCompressorGroup)

Payload sizes to be expected
per compressor context, channel, and interface

```
rohcPayloadSizeTable
  i ifIndex                Integer32(1..2147483647)
  i rohcChannelIndex       RohcChannelIndex
  i rohcCompressorCID      Integer32
  i rohcPayloadSize        Integer32
```

Decompressor Group
(rohcDecompressorGroup)

Decompressor contexts per channels and interface

```
rohcDecompressorTable
  i ifIndex                    Integer32 (1..2147483647)
  i rohcChannelIndex               RohcChannelIndex
  i rohcDecompressorCID            Integer32
    rohcDecompressorState          INTEGER
    rohcDecompressorMode           INTEGER
    rohcDecompressorProfile        Integer32
    rohcDecompressorDepth          Integer32
    rohcDecompressorInPackets      Counter32
    rohcDecompressorOutACKs        Counter32
    rohcDecompressorOutNACKs       Counter32
    rohcDecompressorOutSNACKs      Counter32
```

Statistics Group (1)
(rohcStatisticsGroup)

Outgoing packet counter per header type, compressor context, channel, and interface

```
rohcPacketSizeTable
   i ifIndex                  Integer32(1..2147483647)
   i rohcChannelIndex            RohcChannelIndex
   i rohcCompressorCID           Integer32
   i rohcHeaderIndex             Integer32
     rohcOutPacketCounter        Counter32
```

Statistics Group (2)
(rohcStatisticsGroup)

Incoming packet counter per header type, decompressor context, channel, and interface

```
rohcPacketSizeTable
   i ifIndex                  Integer32(1..2147483647)
   i rohcChannelIndex            RohcChannelIndex
   i rohcDecompressorCID         Integer32
   i rohcHeaderIndex             Integer32
     rohcInPacketCounter         Counter32
```

Statistics Group (3)
(rohcStatisticsGroup)

Error counters per error type,
decompressor context, channel, and interface

```
rohcErrorTable
  i ifIndex                 Integer32(1..2147483647)
  i rohcChannelIndex          RohcChannelIndex
  i rohcDecompressorCID       Integer32
  i rohcErrorIndex            Integer32
    rohcErrorDescr            SnmpAdminString
    rohcErrorCounter          Counter32
```

Discussion Points (1): Channel IDs

- Is the logical hierarchy technically correct?
  - -- interface -- channel -- context ?
  - Is there a chance that the feedback channel uses another interface? (not supported yet)
- What would be an appropriate channel identifier?
  - Is there already a MIB containing channels?

Discussion Points (2): Compressor Context Issues

- Is there a requirement to perform context re-initializations?
  - currently supported
- Is there a requirement to add or remove allowed packet sizes?
  - not supported yet
- Is there a requirement to add or remove payload sizes?
  - not supported yet
- Lifetime of context entries
  - until termination (+timeout) ?
  - until CID re-use?

Discussion Points (3): Statistics

- Are there suggestions for more concrete error types?
- Better having counters per repair strategy instead of per error type?

- Should there be more counters per channel?
  - … and less per context?
  - contexts might be short lived

- Are there ideas for more / less / modified statistics?
  - Packet counter per header type supported
  - Packet counter per profile not supported yet

Discussion Points (4): Conformance

- Which of the groups should be
  - mandatory?
  - optional?
- How to proceed when a MIB for TCP, SCTP, … is required?
  - independent MIBs for each transport protocol?
  - basic module and individual extension modules?
  - open generic approach probably capable of integrating foreseeable future extensions?

Is anyone planning to implement the MIB?

52nd IETF: Agenda (Thu afternoon)

- 1530 TCP
  - 1530 TCP field behavior — West (10)
  - 1540 Requirements document ➡ freeze? — Jonsson (10)
  - 1550 Role of EPIC — West (10)
  - 1600 Progress in merging the drafts — (Authors) (10)
  - 1610 Requirements unmet so far — Jonsson (10)
- 1620 SCTP — Schmidt (20)
- 1640 MIB — Quittek (20)
- 1700 RTP ➡ DS — Chairs (10)
- 1710 Rechartering — (20)

How to get a draft standard?

- Interop, MIBs, Implementer Guide, …
- How to do the surgery?

Proposal:

- I-D outlining the separation Feb 2002
- Do actual surgery once this has stabilized

Rechartering: Goals and Milestones (1)

- **Lower-layer Guidelines: submit for Informational RFC**
  - **WG last-call (again) –03 December 2001**
- **Implementers' Guide: –01 in January**
  - **Align with and feed into DS work**
- **0-byte IP/UDP/RTP**
  - **R-mode LLA**
    - **WG last-call December 2001?**

Rechartering: Goals and Milestones (2)

- Signaling compression
  - Requirements -- I
  - Universal decompressor virtual machine -- PS
  - Protocol/Framework -- PS
  - Signaling compression security analysis – I (later)
  - Example UDVM decompressors – I (IPR, later?)
  - Example extended interactions – I (IPR, later???)
  - If necessary: protocol for extended interactions – PS (IPR)

Work on them now!

Rechartering: Goals and Milestones (3)

- TCP:
  - Requirements and assumptions frozen
    - Call-for-freeze to ROHC, PILC, TSVWG
  - TCP model document: –00 Sep, –01 for SLC (November 2001)
  - draft-ietf-rohc-tcp-00.txt: February 2002
  - WG last-call August 2002, submit September 2002
- EPIC
  - Need to be done before TCP if we want to use it for that
  - Separate notation document draft-ietf-rohc-epic-00: August 2001
  - Decide: Interoperable implementations by Dec 2001?

Rechartering: Goals and Milestones (3a)

- SCTP:
  - Requirements and assumptions frozen: May 2002
    - Call-for-freeze to ROHC, PILC, TSVWG
  - SCTP model document: –00 Mar
  - draft-ietf-rohc-sctp-00.txt: May 2002
  - WG last-call November 2002, submit December 2002
- EPIC
  - Need to be done before SCTP if we want to use it for that
  - Separate notation document draft-ietf-rohc-epic-00: August 2001
  - Decide: Interoperable implementations by Dec 2001?

Rechartering: Goals and Milestones (4)

RTP ROHC Draft standard

- Delineation of framework and profiles ➡ I-D Feb 2002
- MIB
  - draft-ietf-rohc-mib-rtp-00.txt: November 2001
  - WG last-call Apr 2002, submit May 2002
- Draft standard by 3Q2002
  - Separate documents (Framework, 4 profiles): July 2002
  - Merge implementers' guide: July 2002
  - WG last-call August 2002, submit September 2002

Rechartering (5)

Upgrades of RTP ROHC:

- ROHC over reordering channels?
  - Do some of the work in TCP
- UDP-lite profile?
  - Requirements, Specification: I-Ds February 2002
  - WG last-call August 2002, submit September 2002

- Re-recharter Dec 02
- Remember: This all has to go through the ADs…