RObust Header Compression WG (ROHC)

60th IETF
San Diego, August 3, 2004

Chairs:

Carsten Bormann <cabo@tzi.org>
Lars-Erik Jonsson <lars-erik.jonsson@ericsson.com>

Mailing List:

rohc@ietf.org

- **We assume people have read the drafts**

- **Meetings serve to advance difficult issues by making good use of face-to-face communications**

- **Be aware of the IPR principles as stated by RFC 3668**

✓Blue sheets
✓Scribe(s)

60th IETF: ROHC WG Agenda, 1(2)

15:45 - Chair admonishments and agenda	**Jonsson (5)**
15:50 - WG and document status update	**Jonsson (20)**
16:10 - Signaling compression	
16:10 - SigComp implementer's guide issues	**West (15)**
16:25 - Applying SigComp to SIP	**Liu (15)**

60th IETF: ROHC WG Agenda, 2(2)

16:40 - ROHC RTP update

- 16:40 - Implementer´s Guide Jonsson (10)
- 16:50 - Implementation status & feature test list Sandlund (10)

17:00 - ROHC over Channels that can Reorder Packets Jonsson (10)

17:10 - ROHC TCP Pelletier/West/Sandlund (50)

60th IETF: ROHC WG Agenda

15:45 - Chair admonishments and agenda Jonsson (5)

15:50 - WG and document status update Jonsson (20)

16:10 - Signaling compression

- 16:10 - SigComp implementer's guide issues West (15)
- 16:25 - Applying SigComp to SIP Liu (15)

Document status update, 1(3)

- **Old**
  - **RFC 3095: Framework and four profiles** (was: draft-ietf-rohc-rtp-09.txt)
  - **RFC 3096: RTP requirements** (was: draft-ietf-rohc-rtp-requirements-05.txt)
  - **RFC 3241: ROHC over PPP** (was: draft-ietf-rohc-over-ppp-04.txt)
  - **RFC 3242: LLA RTP** (was: draft-ietf-rohc-rtp-lla-03.txt)
  - **RFC 3243: 0-byte RTP req's** (was: draft-ietf-rohc-rtp-0-byte-requirements-02.txt)
  - **RFC 3320: SigComp** (was: draft-ietf-rohc-sigcomp-07.txt)
  - **RFC 3321: SigComp extended** (was: draft-ietf-rohc-sigcomp-extended-04.txt)
  - **RFC 3322: SigComp Req.** (was: draft-ietf-rohc-signaling-req-assump-06.txt)
  - **RFC 3408: LLA R-mode** (was: draft-ietf-rohc-rtp-lla-r-mode-03.txt)
  - **RFC 3409: ROHC RTP LLG** (was: draft-ietf-rohc-rtp-lower-guidelines-03.txt)

Document status update, 2(3)

- New ☺
  - RFC3759: ROHC Terminology & channel mapping examples
  - RFC3816: Definitions of managed objects for ROHC
  - RFC3843: A ROHC profile for IP

- In RFC editor queue
  - NONE!

- Submitted to IESG
  - draft-ietf-rohc-udp-lite-04.txt (PS)

Document status update, 3(3)

- Passed WGLC
  - draft-ietf-rohc-sigcomp-nack-01,
    "A NACK mechanism for SigComp“ (PS)

- Current WG documents
  - RTP/Framework – 2 drafts (impl.guide/interop.status)
  - TCP profile – 5 drafts (req./behavior/profile/notation/repl.)
  - SigComp – 2 drafts (sigcomp-sip/impl.guide)

WG Status, Goals and Milestones

- Slowly, work items are getting completed
- Few active participants, work items are progressed sequentially
- Currently, focus is on ROHC TCP, while the ROHC RTP DS advancement is on hold

WG administrative issues

- Milestones to be updated after IETF 60
- The “Committed WG Document Reviewers” concept has been successful
  - Are now required for all WG document
  - Should be non-authors
  - Must have agreed to follow the document evolution, carefully review the whole document, and respond openly to WGLC
  - Note: Their comments are valued as comments from anyone

60th IETF: ROHC WG Agenda

15:45 - Chair admonishments and agenda — Jonsson (5)

15:50 - WG and document status update — Jonsson (20)

16:10 - Signaling compression

- 16:10 - SigComp implementer's guide issues — West (15)
- 16:25 - Applying SigComp to SIP — Liu (15)

SigComp Implementers’ Guide

Abbie Surtees & Mark West

{abigail.surtees, mark.a.west}@roke.co.uk

Changes (between –02 and –03)

- Clarifications to usage of byte copying rules:
  - Circular buffer usage
    - Additional text / pseudo-code defining use of byte_copy_left / byte_copy_right
  - Things that ***do*** obey byte-copying rules
    - Reading bytes for CRC
    - Reading partial ID for STATE-FREE
  - Things that ***do not*** obey byte-copying rules
    - Reading feedback items
    - Reading state advertisements

Changes (between –02 and –03)

- Duplicate states and STATE-CREATE
  - Time of creation is updated if state is re-created
- DAP (Dummy Application Protocol)
  - Used for first interoperability test at SIPit
  - Protocol description added as appendix
    - Just for reference

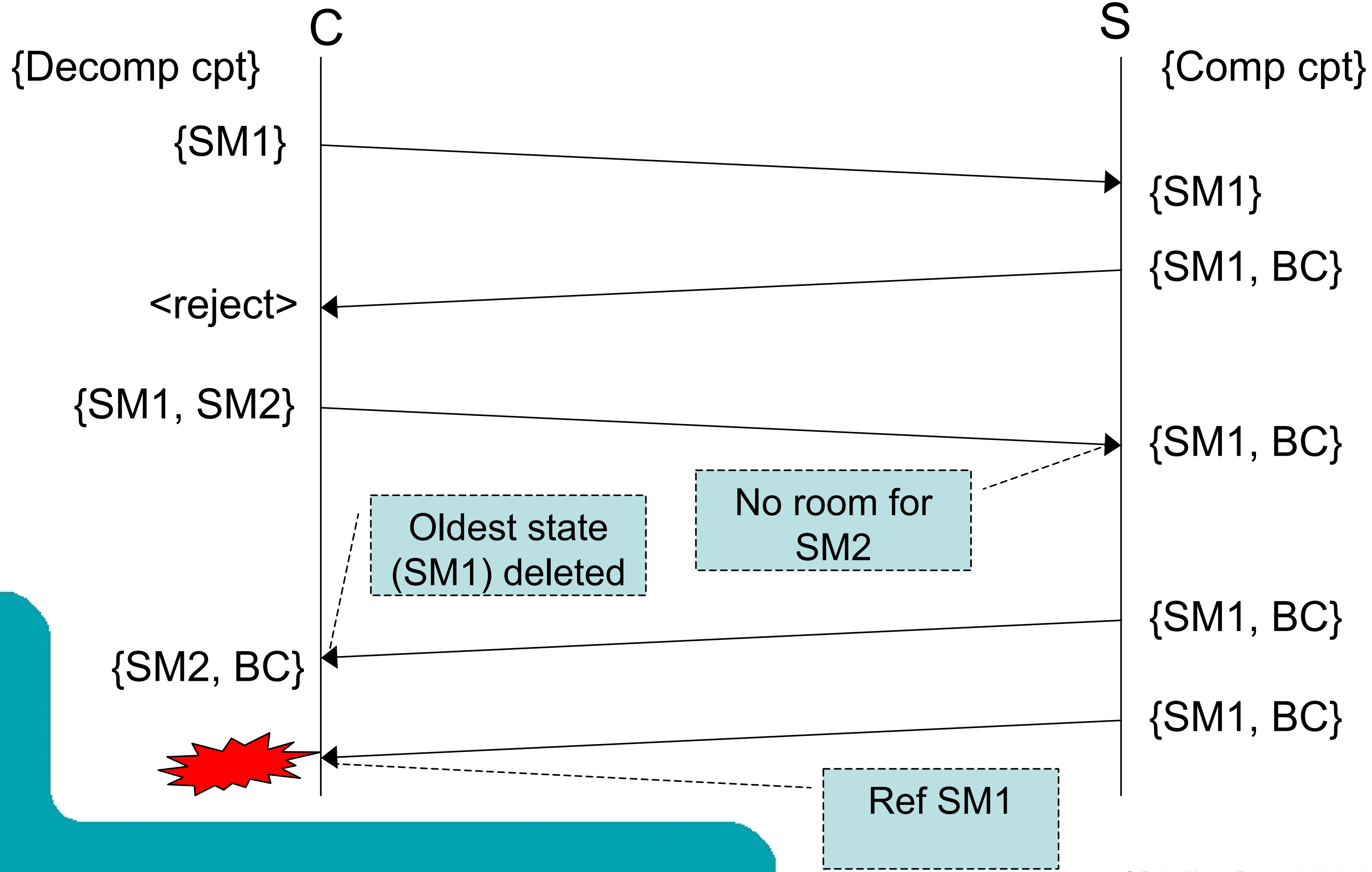
To be added…

- SIP/SDP static dictionary issues (RFC 3486)
  - Slightly broken references to string "application“ (and some others)
  - Point out that compressor should not use references that are inconsistent (in terms of priority) if it would cause a problem
  - But note that this is a compressor-local issue

To be discussed…

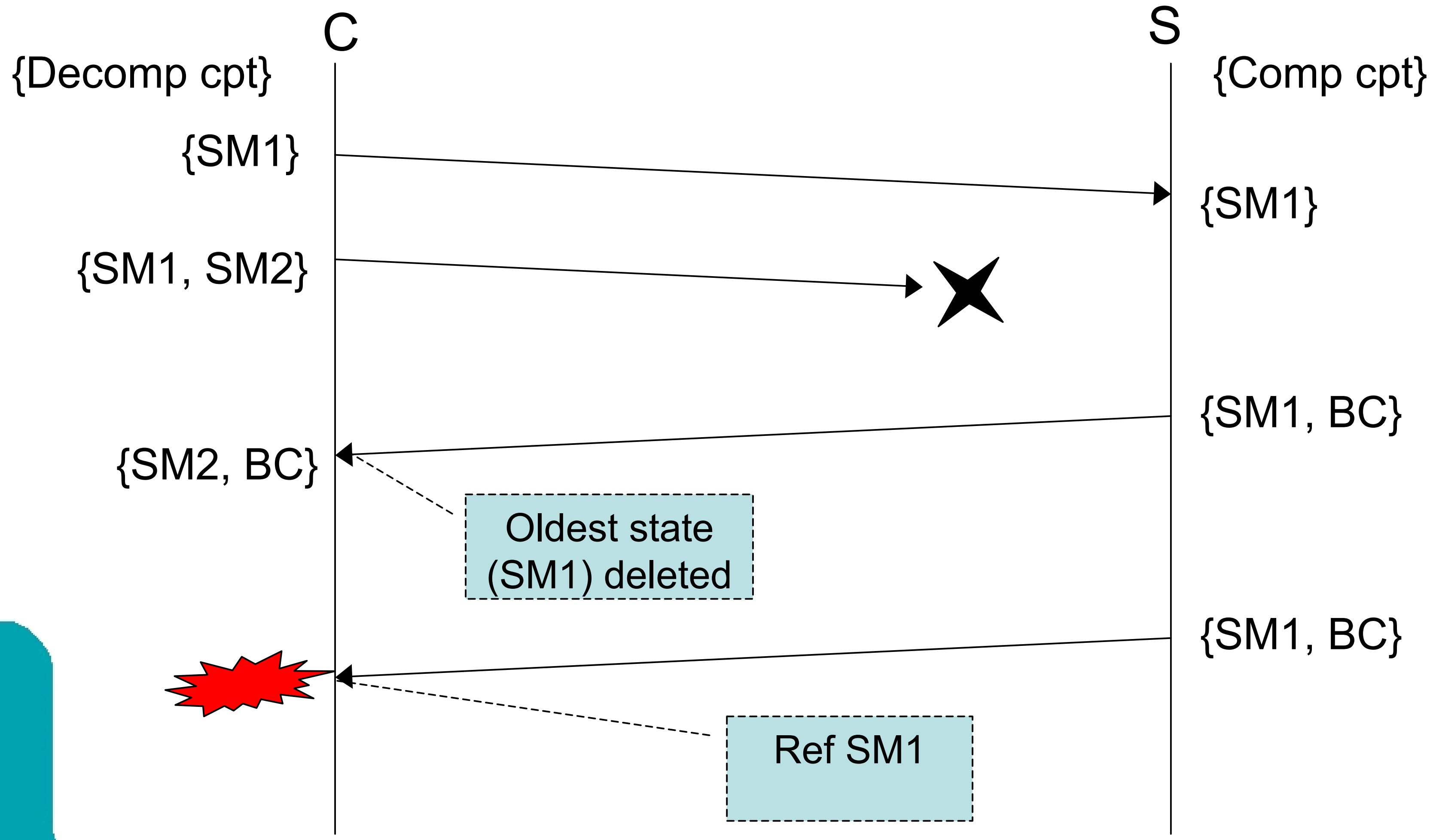
- Shared-mode state inconsistency issue

Shared-mode Issue (1)



© Roke Manor Research Limited 2003

Shared-mode Issue (2)



© Roke Manor Research Limited 2003

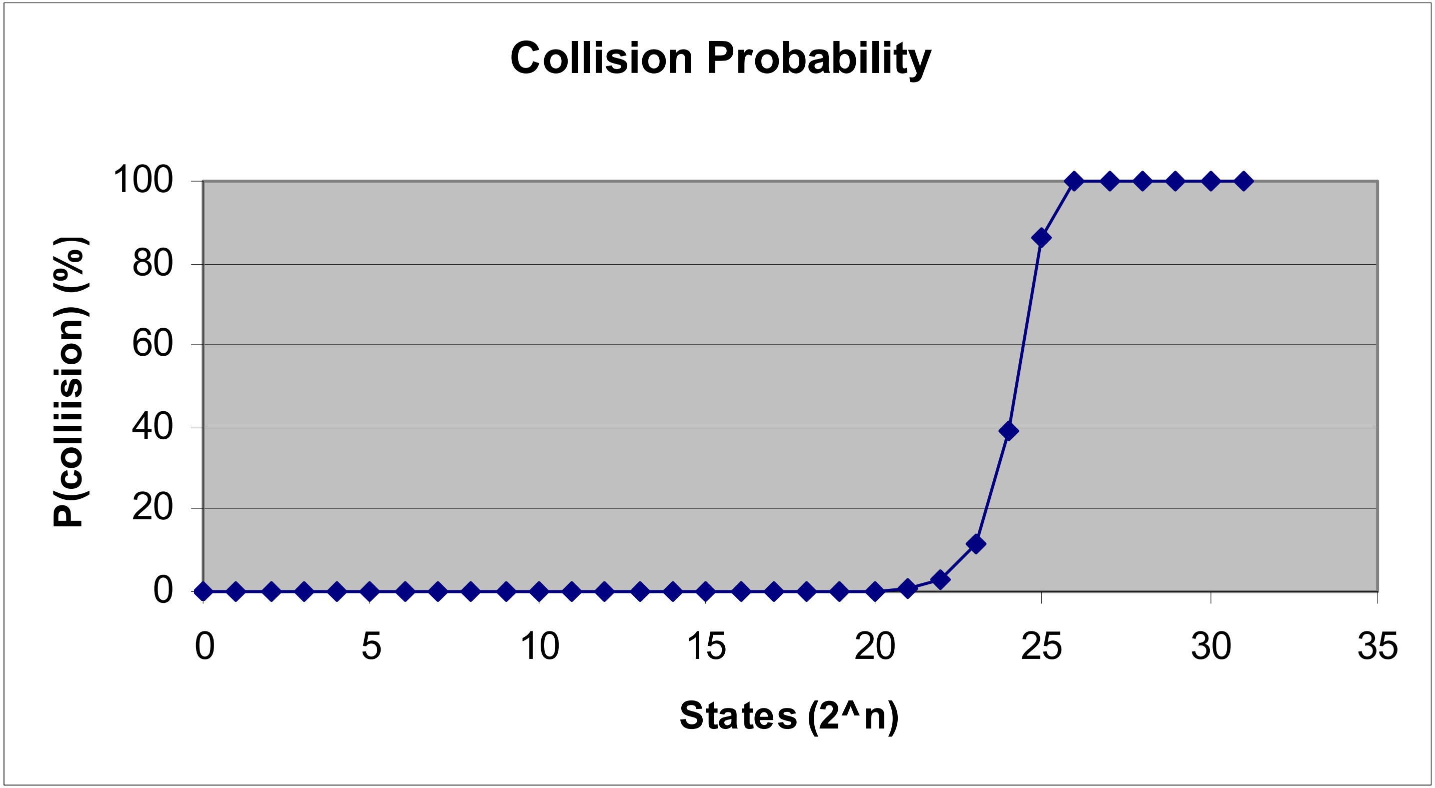
Proposed Solution

- Restrict shared-mode to only attempt creation of *at most one* piece of state
  - Avoids possible ambiguity

A final curiosity…

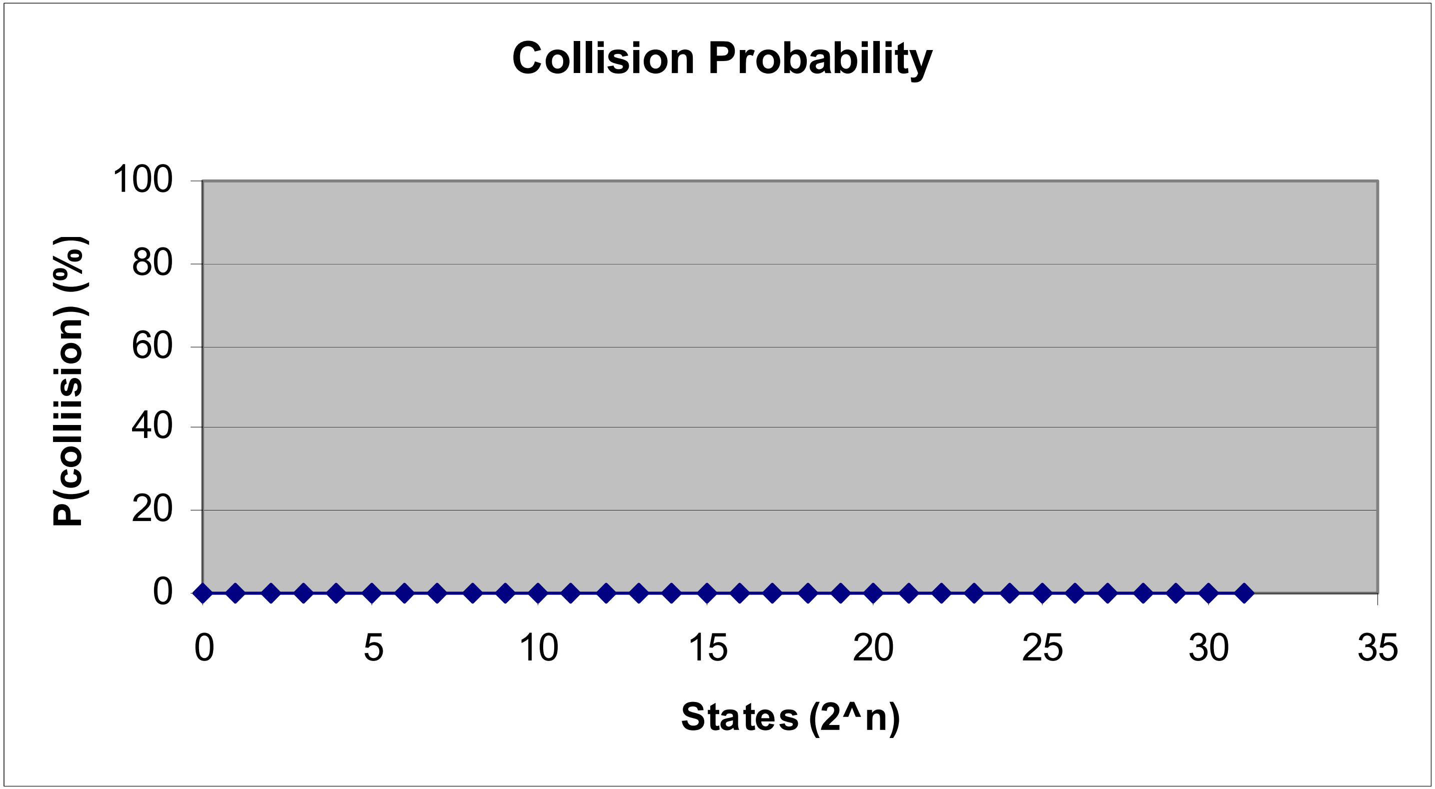
- Calculation of ‘Birthday Paradox’ collision probabilities…
- … or, ‘how long should partial state identifiers be?’
- It is clear that 2^{160} allows for a very large number of states
- But what about 6, 9, 12 byte partial hashes..?

With 6-octet Partial Hash



P(collision) ≈ 1% at 2.4 million states

With 9-octet Partial Hash



P(collision) ≈ 1% at 1×10^{10} states

Questions?

60th IETF: ROHC WG Agenda

15:45 - Chair admonishments and agenda	Jonsson (5)
15:50 - WG and document status update	Jonsson (20)
16:10 - Signaling compression	
16:10 - SigComp implementer's guide issues	West (15)
16:25 - Applying SigComp to SIP	**Liu (15)**

SigComp-SIP

Zhigang Liu

IETF-60

(Aug 3, 2004)

Minimum Values of SigComp Parameters

- DMS (decompression_memory_size): 8 KB
  - Per message
  - Looks we reached agreement in the mailing list
- SMS (state_memory_size): 2KB
  - Per compartment
  - Question: should it be lowered to 0KB?
    - Pro: allow SigComp for stateless SIP proxies
    - Con: take 1 more RTT to upload state other than decompression bytecode
- CPB (cycles_per_bit): 16
  - No comments so far

Compartment and State Management

- Per dialog compartment
  - This is the current text, but is it too short lived?
  - What about SIP messages out of dialog?
- Per registration compartment
  - This is the 3GPP approach and likely the PoC approach in OMA
- Long-life state
  - The receiver of a state offers that the state lives longer than the compartment in which it was created
  - The receiver specifies the life of the state
  - Should be “loosely clocked”, i.e., leave enough margin for clock errors and transmission delay jitter

Multiplexing SigComp and Plain SIP Messages on the Same TCP Connection (1/4)

- “multiplex” in RFC 3320 section 3.1 does not mean real multiplex
    - “All SigComp messages contain a prefix (the five most-significant bits of the first byte are set to one) that does not occur in UTF-8 encoded text messages [RFC-2279], so for applications which use this encoding (or ASCII encoding) it is possible to multiplex uncompressed application messages and SigComp messages on the same port. Applications can still reserve a new port specifically for SigComp however (e.g., as part of the discovery mechanism).”
- So, can we add real multiplexing to SigComp? It means a SIP sender can send either SigComp or plain SIP messages on the same TCP connection in any way it wants.
- Or, is it too luxury to have?

Multiplexing SigComp and Plain SIP Messages on the Same TCP Connection (2/4)

- Multiplexing offers flexibility and solves multiple problems.

- Problem #1: when a SIP sender does not know whether the receiver supports SigComp, it has to start with sending plain SIP messages and then sends SigComp messages.
    - Alternative solutions:
        - Open a new TCP connection
            - add additional delay, e.g. 2+ seconds in GPRS or low-bandwidth links in general
            - for some systems (e.g. 3GPP), need to close existing connection and re-open with the same port number
            - violates RFC 3261 in some RFC3486 cases: a response must be sent over the same TCP connection from which the request was received.
            - Issues described in draft-ietf-sip-connect-reuse-02.txt
        - TLS-like one-way transition from non-SigComp to SigComp
            - How does it work? Why is it better than multiplexing?

Multiplexing SigComp and Plain SIP Messages on the Same TCP Connection (3/4)

- Problem #2: 64KB limit on SigComp message. The sender starts SigComp then needs to send a SIP message > 64 KB.
  - Alternative solutions:
    - Close existing TCP connection and reopen a new one.
      - Rationale: it takes a while to transmit the message anyway. So who cares the extra delay?
      - But what if after the >64KB message, there is a sequence of small SIP messages that need to be compressed?
- Problem #3: allows the sender to switch between SigComp and plain messages to save CPU/memory
  - Alternative solution: what about just opening two TCP connections simultaneously?
    - One more TCP context ➔ cost more memory on both client and server
    - Again, issues raised in draft-ietf-sip-connect-reuse-02.txt.
- Problem #4: fallback to plain message when SigComp decompression failed due to reasons other than state loss (e.g. UDVM failure). The sender wants to bypass SigComp completely.
  - Or, are we too paranoid here?

Multiplexing SigComp and Plain SIP Messages on the Same TCP Connection (4/4)

- Summary:
  - Multiplexing is one solution that solves all the problems without increasing the number of TCP connections.
  - The main argument against it is complexity. For each problem, there may be a simpler solution. However, this is still under discussion. Also, it requires extra interaction between SigComp and SIP.
  - This is also an issue for other application protocols. SIP is just where we begin.

Other Issues

- Continuous mode
  - Should it be allowed if we have security below transport (e.g. Ipsec)?
  - So far, it seems nobody cares.

- When SIP must use TCP, instead of UDP?
  - RFC3261: “If a request is within 200 bytes of the path MTU, or if it is larger than 1300 bytes and the path MTU is unknown, the request MUST be sent using an RFC 2914 [43] congestion controlled transport protocol, such as TCP.”
  - Now, with SigComp in picture, should the above limits apply before SigComp or after?

60th IETF: ROHC WG Agenda

16:40 - ROHC RTP update

- **16:40 - Implementer´s Guide** **Jonsson (10)**
- 16:50 - Implementation status & feature test list Sandlund (10)

17:00 - ROHC over Channels that can Reorder Packets Jonsson (10)

17:10 - ROHC TCP Pelletier/West/Sandlund (50)

ROHC RTP Implementer's Guide, 1(3)

- draft-ietf-rohc-rtp-impl-guide-05.txt
- News in the updated version
  - Rewritten 4.4, "TS_STRIDE and the Tsc flag in Extension 3"
  - Clarified IP-ID encoding (based on RND) in section 7.2
  - New 7.6, clarifying multiple CRC options in feedback
  - Open issues in new Appendix
- Open issues
  - Mode inheritance when a context is re-used
  - Slope used to compress/decompress RTP Timestamp

Mode inheritance when re-using a context

- The initial observation was that it is not clear whether operational mode is inherited when re-using a context

- After discussions, it was proposed that
  - When re-using a context with the same profile, mode should be kept
  - When re-using a context with a different profile, mode should simply be set to intial mode at both compressor and decompressor

- See mail list discussion with subjects “ROHC context reuse & mode” and “Context reuse revisited” from March-June 2004

- This issue has not yet been closed, please contribute!

Slope used to compress/decompress TS

- On March 16th, a question entitled ” Slope used to compress/decompress RTP Timestamp field” was sent to the ROHC mail list

- This triggered a long discussion that continued during March and April, but no agreeable solution was found

- It was a complex discussion based on various views on the intentions when writing RFC 3095 and what is actually defined/described in RFC 3095, as well as opinions on what would make sense technically

- This issue has not yet been closed, please contribute!

60th IETF: ROHC WG Agenda

16:40 - ROHC RTP update

16:40 - Implementer´s Guide Jonsson (10)

16:50 - Implementation status & feature test list Sandlund (10)

17:00 - ROHC over Channels that can Reorder Packets Jonsson (10)

17:10 - ROHC TCP Pelletier/West/Sandlund (50)

RFC3095 Implementation status

IETF-60, San Diego, 3/8 2004

Kristofer Sandlund, Effnet AB

ROHC Interop test 5

- Held in Stockholm, Sweden, June 2004
- Participants: Effnet, Ericsson
- Focus:
  - Test the entire UDP profile
  - Test all packet types, extensions and semi-static field changes in RTP profile
  - ROHC Segmentation
  - Piggybacked feedback

EFFNET

Results of Interop 5

- No new problems with the specification encountered.
- Test focus successfully interoperated:
  - All packet formats, extensions and changes in semi-static fields (except list compression related parts). Applies to both RTP and UDP profile.
  - Piggybacked feedback ok
  - ROHC segmentation test scenarios ok
  - Some streaming voice and video tested
- Draft updated (draft-ietf-rohc-rtp-rfc3095-interoperability-03.txt)

EFFNET

Untested features

- CSRC list compression (some basic tests done at interop 4 in Berlin, but most of it remains)
- Extension header list compression (completely untested)
- Timer-based RTP compression
- Some feedback options (REJECT etc)
- ROHC ESP profile
- CID reuse (as per list discussion, no final decision taken?)
- IR without dynamic chain/payload
- Anything else that I've forgotten?

Questions?

EFFNET

60th IETF: ROHC WG Agenda

16:40 - ROHC RTP update

16:40 - Implementer´s Guide Jonsson (10)

16:50 - Implementation status & feature test list Sandlund (10)

17:00 - ROHC over Channels that can Reorder Packets Jonsson (10)

17:10 - ROHC TCP Pelletier/West/Sandlund (50)

ROHC over Channels that can Reorder Packets, 1 (2)

- Individual draft (Pelletier/Jonsson/Sandlund):
  - draft-pelletier-rohc-over-reordering-00.txt

- Why this document?
  - Questions about ROHC and reordering channels are common
  - As ROHC explicitly states the inability to operate over channels that can reorder packets, people tend to think ROHC is particularly unsuitable for such scenarios, compared to other HC alternatives, which is not true

ROHC over Channels that can Reorder Packets, 2 (2)

- What's in the document?
  - A problem description for ROHC and reordering channels
  - Suggestions for how to implement RFC3095-based profiles over reordering channels
  - Ideas for how existing profiles could be updated and how new profiles can be defined to efficiently cope with reordering

- What now?
  - Please read the document and help us improve it
  - Should we make this a WG document to be published as INF?

60th IETF: ROHC WG Agenda

16:40 - ROHC RTP update

16:40 - Implementer´s Guide Jonsson (10)

16:50 - Implementation status & feature test list Sandlund (10)

17:00 - ROHC over Channels that can Reorder Packets Jonsson (10)

17:10 - ROHC TCP Pelletier/West/Sandlund (50)

60th IETF – ROHC WG

Status of the ROHC-TCP/CR/FN work

ghyslain.pelletier@ericsson.com
+ 46 8 404 29 43

Status of the ROHC-TCP work

- Drafts Status:

draft-ietf-rohc-tcp-requirements-07.txt	(Updated June 04)
draft-ietf-rohc-tcp-field-behavior-03.txt	(Expired)
draft-ietf-rohc-tcp-07.txt	(Updated July 04)
draft-ietf-rohc-context-replication-03.txt	(Updated July 04)
draft-ietf-rohc-formal-notation-03.txt	(Updated July 04)

- Work has progressed a lot, nearing completion

Recent changes: draft-ietf-rohc-tcp-07.txt

New packet format definitions :)	
Definition of irregular chains	(Section 4.5)
Clarifications on List Encoding for TCP options	(Section 4.6.1)
Definition of Item Table mappings for TCP options	(Section 4.6.2)
Replication and TCP options	(Section 4.6.3)
Clarifications on Extension Headers	(Section 4.6.5)
ACKing IR packets requires CRC feedback option	(Section 5.2.7)
MSN initialization and re-initialization	(Section 6.2)
General format of the ROHC-TCP compressed hdr	(Section 6.5.3)

Recent changes: draft-ietf-rohc-tcp-07.txt (8)

- General format of the ROHC-TCP compressed hdr (Section 6.5.3)

```
  0   1   2   3   4   5   6   7
 --- --- --- --- --- --- --- ---
:         Add-CID octet         :  if for small CIDs and CID 1-15
+---+---+---+---+---+---+---+---+
|   first octet of base header  |  (with type indication)
+---+---+---+---+---+---+---+---+
:                               :
/   0, 1, or 2 octets of CID    /  1-2 octets if large CIDs
:                               :
+---+---+---+---+---+---+---+---+
/   remainder of base header    /  variable number of bits
+---+---+---+---+---+---+---+---+
:                               :
/ Header Chain Irregular Part   /  variable (see section 6.6.9)
:                               :
 --- --- --- --- --- --- --- ---
:                               :
/  TCP Options Irregular Part   /  variable (see section 6.6.5)
:                               :
 --- --- --- --- --- --- --- ---
```

Irregular fields of tunneling headers and extension headers

Recent changes: draft-ietf-rohc-tcp-07.txt (3)

- Clarifications on List Encoding for TCP options (Section 4.6.1)

- Initialization -> complete representation w/ option contents sent
- When nothing changes, nothing is sent
- When structure constant, content changes -> content to irregular chain
- Structure changes -> compressed representation of list is sent

Recent changes: draft-ietf-rohc-tcp-07.txt (4)

- Definition of Item Table mappings for TCP options (Section 4.6.2)

- **These mappings are used for items that occur frequently**
- **Option type can be omitted from compressed packet**
- **Option type is derived based on index number**

Option name	Table index
NOP	0
EOL	1
MSS	2
WINDOW SCALE	3
TIMESTAMP	4
SACK-PERMITTED	5
SACK	6
Generic options	7-15

Recent changes: draft-ietf-rohc-tcp-07.txt (5)

- Replication and TCP options (Section 4.6.3)
  - TCP options are replicated
  - A generic compressed list is then sent to update the content of the item table to correspond to the new flow.

Recent changes: draft-ietf-rohc-tcp-07.txt (7)

- MSN initialization and re-initialization (Section 6.2)

MSN is reinitialized only for the 1st TCP flow associated to a CID

I.e. it is reused if multiple flows are compressed one after the other using same CID.

Changes: draft-ietf-rohc-context-replication-03.txt

- **Inter-profile context replication is not defined anymore**
  - replication between profiles will not be supported
  - text moved to appendix as guidelines for future update
- **Requirements for selection of base context**
  - added requirement that the context selected must have been acked with the CRC option.

    (to protect against erroneous feedback, or feedback in-flight while compressor was reinitializing)
- **Additional decompressor feedback logic**
  - should use the CRC option when acking an IR or IR-CR packet

Next Steps

draft-ietf-rohc-tcp-requirements-07.txt	Ready for wglc?
draft-ietf-rohc-tcp-field-behavior-03.txt	Expired, resubmit Ready for wglc?
draft-ietf-rohc-tcp-07.txt	Update and resubmit Last round before wglc?
draft-ietf-rohc-context-replication-03.txt	Ready for wglc?
draft-ietf-rohc-formal-notation-03.txt	Update and resubmit Last round before wglc?

Next Steps for ROHC-TCP/FN?

We need 2 committed reviewers for each document!

Quick updates and wglc for remaining documents
-> beginning of September?

ROHC-TCP Packet Format Overview

IETF-60, San Diego, August 2004

Kristofer Sandlund, Effnet AB

EFFNET

Static/Dynamic/Irregular Chains

- Static chains and Dynamic chains using the same concepts as in RFC3095 (these are used in IR/IR-DYN packets).
- Irregular chain is an all-new conecpt. This is used in compressed headers for the fields that must always be sent uncompressed. The irregular chain is used to construct a "general compressed packet format" as in RFC3095. All headers have irregular chains (can be empty)

Multiple IP Headers

- Multiple headers handling is simplified by the use of irregular chains so that only the innermost IP header and TCP header must be included in the base format of compressed headers
- Innermost IP header (v4 or v6) uses different irregular chains, since its dynamic fields are included in the base compressed header format
- The IP-ID must always be sent in full (unless constant zero) for all but the innermost header (since correlation to MSN is weaker for tunnel headers). Innermost IP-ID have NBO,RND,ZERO flags as in the IP-only profile

Extension Headers

- Not compressed using list compression
- Extension headers are treated as all other headers, having static/dynamic/irregular chains
- Header chain structure is static (all next header fields are in static chain), since insertion/deletion of headers is not expected without making other major changes to the flow
- ESP NULL - No trailer compression

TCP Options

- These are compressed with List Compression type 0 from RFC3095 with restrictions
- Max 16 items in table, max 15 in a list, static mappings of frequently occuring options
- Each type of item has a "list_item" format which is the data appearing in compressed lists
- TCP options have irregular chains
- Replicating options mean all the table is replicated in context and a list is sent for present options
- SACK and EOL format get a bit complex, the rest are quite simple
- Generic option for handling future & rare TCP options

TCP Compressed Packet Formats

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

TCP Compressed Packet Formats

- Basic structure
  - Two sets of ‘*CO*’ packet formats
    - Sequential IP-ID
    - Non-Sequential IP-ID (random and zero) (includes IPv6)
  - ‘Common’ packet format
    - Switching between packet formats
    - Handling of ‘rare’ change patterns
    - ‘Catch-all’ for compressed packets
      - Serves similar role to UOR-2 + Ext. 3 (for fans of RFC 3095)

TCP Compressed Packet Formats

- Where did the formats come from?
  - We made them up ☺
- Analysed a number of bulk and interactive TCP flows
- ‘normal’ link
  - Moderate round-trip time (~100ms)
  - Moderate packet loss
- Capture the ‘most common’ change patterns

Compressed Format Summary

- Currently have
  - 18 packet formats for ‘sequential’ IP ID
    - CO packets are in the range 6 – 11 octets
  - 12 packet formats for ‘random’ IP ID (includes ‘zero’)
  - 1 common packet format
- Target is for approx. 12 formats for each set
  - But also trying to get sensible coverage of ‘most likely’ change patterns
- Note that unlike RFC 3095, there are no extension headers

Static-Chain representation

- Slight change from RFC 3095
  - Use a single bit to represent IPv4/IPv6
  - Frees up some other bits
    - Can indicate zero flow-label as a special case…
    - And squeeze an extra byte out of the IPv4 chain

TCP Context Replication

- Watchword is simplicity…
  … honest
- Replication is only currently defined between TCP contexts
- Aim of replication
  - Elide fields where possible
  - Provide minimal set of options otherwise
    - e.g. item is either wholly present or not
  - Extension headers are typically ‘the same’ or ‘carried in full’
- More complex ideas have been proposed
  - Many of these are quite clever
  - However, gain is seen as relatively small compared to increase in complexity

Extension Headers

- What can we handle?
  - IPv6 ‘destination option’ headers
  - IPv6 ‘hop-by-hop option’ headers
  - IPv6 ‘routing’ header
  - GRE tunnel header
  - IP ‘minimal encapsulation’ header
  - IPsec AH
  - IPsec ESP (with *NULL* encryption)
- And a near-arbitrary number of these…
  - … see also the ‘IP only’ profile for ROHC

TCP Options

- SYN-only options are specially handled by ‘pre-loading’ of the list table entries
- Most options can be compressed
    - SACK
    - SACK Permitted
    - WScale
    - Timestamp
    - MSS
- And allow for ‘generic’ options that remain static

Some specific issues

- Sequence / Acknowledgement Numbers
  - Major issue for TCP compression
  - General approach is to provide ‘enough’[1] bits
  - Also use scaling property, where possible
    - If sequence (or ack) changes by payload size for a number of consecutive packets, scale by this

[1] Obviously we will never agree on how many bits are ‘enough’, so we have tried to provide more than necessary for Ethernet. We don’t want to make life hard for big MTUs, but do you *really* need (optimal) header compression..?

Some specific issues

- ECN and reserved bits
  - IP and TCP ECN bits + TCP reserved bits
  - Treated together
  - Either always present or set to 0
  - No special handling for IP reserved bits
- TCP Window
  - Analysis shows that more sophisticated handling may be possible
  - But additional complexity does not give obvious benefit
  - So stick with fairly simplistic ‘LSB’ handling

Some specific issues

- MSN bits
  - Generally increased number of bits in compressed packets
  - See also ‘re-ordering’ draft
    - Allows RoHC TCP to handle a limited amount of channel re-ordering
    - Requires MSN bits in every packet
      - Not that much of an overhead, though
- CRC
  - 3-bit for sequence-number updating packets
  - 7-bit for more ‘complex’ packets

IPR

- Hopefully IPR isn’t much of an issue?
  - At least *we* think it is under control
- But if anyone knows otherwise…

Questions?