RObust Header Compression WG (ROHC)

56th IETF
San Francisco, March 17, 2003

Chairs:

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

Mailing List:

rohc@ietf.org

56th IETF: Pre-Agenda

- WG chair admonishments
- Real agenda

✓Blue sheets
✓Scribe(s)
✓Jabber?

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)
  - Working Group Chairs: Lars-Erik Jonsson, Carsten Bormann
  - Technical Advisor: Erik Nordmark
- **Work is done on email list: rohc@ietf.org**
  - And on IETF meetings, interim meetings, informal meetings
  - Mailing list is official channel, though

Purpose of WG meetings

- We assume people have read the drafts
- Meetings serve to advance difficult issues by making good use of face-to-face communications
- No technical presentations, but lead-ins to stimulate/elicit discussion
- Objective: generate technical consensus

RFC 2026: Internet Standards Process

- **Standards track RFCs:**
  - **WG consensus (as judged by WG chairs)**
  - **WG last call**
  - **IETF last call**
  - **IESG approval (based on AD recommendation)**
    - Quality control!
  - **Publication by the RFC Editor**
- **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!

56th IETF: ROHC WG Agenda, 1(2)

19:30 - Chair admonishments and agenda	Jonsson (10)
19:40 - WG and document status update	Jonsson (20)
20:00 - Signaling compression	
20:00 - SigComp Feedback	Roach (15)
20:15 - SigComp interoperability report	West (10)
20:25 - The SigComp binary multiplexing issue	Price (5)

56th IETF: ROHC WG Agenda, 2(2)

20:30 - RTP update	
20:30 - Implementer´s Guide	Jonsson (5)
20:35 - Implementation status and test list	Jonsson (3)
20:38 - "IP-only" profile	Jonsson (2)
20:40 - Formal HC notation	Bormann/Ozegovic/Price (60)

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)
- **New ☺**
  - **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 related
  - RFC3485: SigComp SIP/SDP static dictionary
  - RFC3486: Compressing SIP with SigComp
- In RFC editor queue
  - NONE!
- Submitted to IESG
  - draft-ietf-rohc-mib-rtp-06.txt (PS)
  - draft-ietf-rohc-terminology-and-examples-02.txt (Info.)

Document status update, 3(3)

- Current WG documents
  - RTP/Framework – 3 drafts
  - TCP profile – 4 drafts
  - SCTP profile – 1 draft (requirements) Work on hold
  - Notation – 1 draft

WG Status, Goals and Milestones

- **WG is progressing slowly, but still progressing**
- **Two major work items are late**
  - **ROHC RTP DS Advancement**
  - **ROHC TCP**

WG administrative issues

- **Milestones to be updated after IETF 56**

- **Introducing new review concept: "Committed WG document reviewers"**
  - **Will be required for each WG document**
  - **Designated as such by chairs, preferably on an early stage**
  - **Should be non-authors**
  - **Must have agreed to follow the document evolution, carefully review the whole document, and respond openly to WGLC**
  - **Will be acknowledged separately in the document**
  - **Note: Their comments are valued as comments from anyone**

WG items not covered by the agenda, 1(4)

- ROHC TCP Profile
  - Profile draft not updated since November
  - Update on its way to include context replication, etc
  - Behavior document updated (yet another will soon follow)
  - Further TCP work dependent on the formal notation work

WG items not covered by the agenda, 2(4)

- **Modified RTP/UDP profiles for UDP-Lite compression**
  - **UDP-Lite is a slightly modified UDP (length->coverage)**
  - **Individual draft available**
    - Analyzing the differences between UDP and UDP-Lite
    - Discussing implications for compression
    - Proposing modifications to derive new UDP-Lite profiles fro the existing RTP/UDP profiles
  - **Plan to request resubmission as WG draft**
    - Read and contribute!

WG items not covered by the agenda, 3(4)

- SCTP compression in the ROHC charter, 1(2)
  - Added to the charter when WG activity was high
  - Added as a prediction of future expectations rather than an outcome of well justified immediate requirements
  - We have repeatedly asked for interest and contributions for SCTP header compression, but not received a single response

WG items not covered by the agenda, 4(4)

- **SCTP compression in the ROHC charter, 2(2)**
  - **Intention to remove it from the charter communicated by the chairs to the mailing list in January**
  - **To keep it in the charter, we asked for**
    1) justifications for developing SCTP header compression now
    2) industry support
    3) commitments for doing the actual work
  - **Answers might potentially be considered as 3), but we believe such a significant work item would have to require also 1) & 2)**

56th IETF: ROHC WG Agenda, 1(2)

19:30 - Chair admonishments and agenda	Jonsson (10)
19:40 - WG and document status update	Jonsson (20)
20:00 - Signaling compression	
20:00 - SigComp Feedback	Roach (15)
20:15 - SigComp interoperability report	West (10)
20:25 - The SigComp binary multiplexing issue	Price (5)

SigComp NACK: Changes

Adam Roach

3/17/2003

Moved NACK information

- Now at end of message
- Allows NACK to be distinguished from standalone feedback

Changed TCP Behavior

- Reception of NACK now looks like transport-level failure to application
- No longer requires SigComp layer to handle retransmission of corrupted messages

Added NACK detection

- Can now detect whether other endpoint supports NACK
- Uses LS bit of byte 11 to indicate NACK support
- Reserves remainder of bytes 10 and 11 for future bit-flags
- Based on byte 11, bytecodes can include indication in feedback
- *Side Note: Should we IANA register the first 32-bytes of UDVM memory space?*

Five New Reason Codes

- INVALID_OPERAND
- ID_TOO_SHORT
- ID_NOT_UNIQUE
- MULTILOAD_OVERWRITTEN
- STATE_TOO_SHORT

Non-normative editorial changes

- Revised server failover motivation
- Updated references to new RFCs
- Handful of tiny changes

What Now?

- More eyeballs needed; feedback has been sparse.
- Consensus on adding to charter?
- Would be nice if someone else implemented it so we could demonstrate interoperability

56th IETF: ROHC WG Agenda, 1(2)

19:30 - Chair admonishments and agenda	Jonsson (10)
19:40 - WG and document status update	Jonsson (20)
20:00 - Signaling compression	
20:00 - SigComp Feedback	Roach (15)
20:15 - SigComp interoperability report	West (10)
20:25 - The SigComp binary multiplexing issue	Price (5)

SigComp
Interoperability Test @ SIPit

Mark West

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

What was the point?

- Primarily to test SigComp
  - (with or without SIP)
- What does that really mean..?
  - VM and SigComp exercised by Torture Tests…
  - … but no substitute for what we *really* do!
- So what did we test?
  - Tried to show that all Decompressor/UDVM end-points behave consistently
  - Also allows testing of compressor implementations
    - And assumptions that compressors make…

Implementations

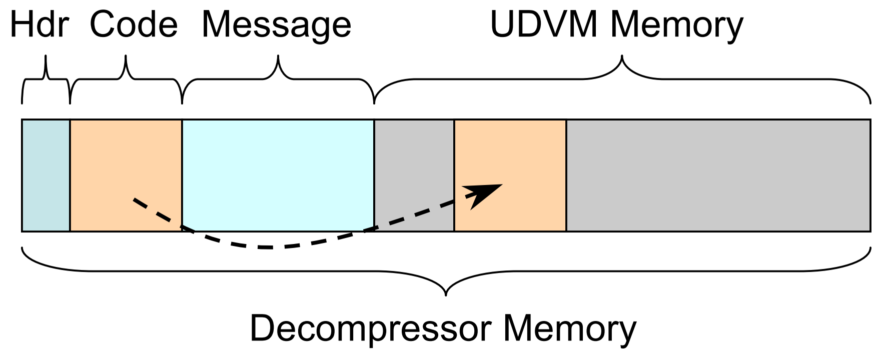
- Impressive turn-out of 6 implementations:
  - Dynamicsoft
  - Ericsson
  - Institute for Communications Research Singapore
  - Lab for Telecommunications Sciences
  - Nokia
  - Siemens/Roke Manor

The Good

- It worked!

The Bad

- Decompressor memory / UDVM memory
  - For UDP, the UDVM has the remaining memory after the decompressor has stored the message
  - If the message includes bytecode, this appears twice in the Decompressor memory
  - Makes bootstrapping very difficult
    - Especially since first message often compresses less well



The Bad (cont.)

- Reading past the end of the input
  - If INPUT-BYTES (or INPUT-BITS) is called when there is no more input, it returns no data
  - *But*, if there are 6 bytes left and we ask for 12, should 0 or 6 bytes be returned?
    - Most interpreted RFC 3320 to say ‘0’
    - However, at least one chunk of bytecode relied on the answer being ‘6’
  - At least clarification is required:
    - If 6 bytes are left and 12 are requested, what happens to the 6 bytes?
      - Are they left unchanged in the input buffer?
      - Are they discarded?

The Bad (cont.)

- Should STATE-FREE check the minimum access length?
  - If not, an attacker can delete state with a shorter hash than the minimum access length
- Cycle counting
  - Didn’t break anything
  - See later!

The Ugly

- State retention priority
  - 65535 < 0 < 1 < … < 65534
- Byte-copying
  - END-MESSAGE is listed in §8.4 covering byte-copying rules
  - STATE-CREATE is not, but clearly should obey the rules
- I-bit
  - *only* affects advertisement of state items
  - Many implementations wire these into the byte-code, so advertisements are received even if the I-bit is set
- S-bit
  - Enough said…

The Ugly (cont.)

- Cycle counting
  - Several implementations had incomplete cycle-counting code
  - There were cases of inputs decompressing on one VM and failing on another…
    - Ok, provided that decompressor is generous (an implementation can never give *less* cycles than it should)
  - Can be ‘entertaining’ for compressor to verify that a message *will* decompress ok

Odds and Ends

- There is no torture test for byte_copy_left, however there is also no problem with its definition!
- Some Dummy Application Protocol (DAP) issues [just in case we ever use it again!]
  - Compartment ID is an integer
  - Implementations should accept (more or less) arbitrary whitespace in the header
  - Some confusion about when to wait for responses and what to wait for (limits complexity of tests)
  - Definition of opening and closing compartments is unclear

Compressors

- A variety of compression algorithms were used, which was encouraging:
  - All appeared to be Lempel-Ziv based
  - Some like LZ77, others more *deflate*-like
  - Various extensions for, e.g. literal encoding; static dictionary access, etc.

Quick Summary of Testing

- Many instructions utilised in the tests
  (but not all…)
- State handling was reasonably well exercised
- Various buffer sizes and responses to advertisements were tested
- Compression with and without static dictionary
- Testing was done with and without feedback
- Testing with multiple applications (SIP and DAP)
- We didn’t try and attack anything
- (Probably should do a more detailed break-down at some point..!)

Overall

- For a first test, good degree of interoperability was achieved
- Reasonable coverage of tests
- Always going to be an issue with testing the rarer instructions / options
- Need to be clear about what is SigComp behaviour and what is Application behaviour
- ‘Distributed’ testing is easier than with ROHC-RTP, especially now we know what we’re doing!
  - Blatant plug: http://sigcomp.srmr.co.uk
- Overall, it’s looking good…

56th IETF: ROHC WG Agenda, 1(2)

19:30 - Chair admonishments and agenda	Jonsson (10)
19:40 - WG and document status update	Jonsson (20)
20:00 - Signaling compression	
20:00 - SigComp Feedback	Roach (15)
20:15 - SigComp interoperability report	West (10)
20:25 - The SigComp binary multiplexing issue	Price (5)

Multiplexing in SigComp

Richard Price

Roke Manor Research

(richard.price@roke.co.uk)

© Roke Manor Research Limited 2002

SIEMENS

Requirements on Application

Roke Manor Research

- SigComp offers a transport service to applications
  - Using SigComp requires more than just an “on” switch!
  - For security, some decisions are left to the application
- Applications using SigComp must provide:
  - Mechanism for SigComp discovery
  - Method for recognising SigComp messages
  - Classification of messages into compartments
  - Compartment set-up and teardown
  - Decision on whether to use “continuous mode”
  - More default memory or UDVM cycles

© Roke Manor Research Limited 2002

SIEMENS

Multiplexing Messages

Roke Manor Research

- SIP messages can contain binary data
  - May be difficult to compress (e.g. JPEGs)
  - Messages can be long (> 64 Kbytes)
- “Uncompressible” messages can be sent as-is
  - Application and SigComp messages can be multiplexed
  - Over UDP this is easy!
- Over TCP some extra issues must be considered
  - How to determine when an application message ends?
  - Interaction between application and SigComp delimiters

© Roke Manor Research Limited 2002

SIEMENS

Open Issues

Roke Manor Research

- Is this multiplexing mechanism useful?
  - Intent was to support “SigComp-unaware” endpoints
  - SigComp can easily handle uncompressible messages
  - Why not just use SigComp on a per-connection basis?
- Should “continuous mode” be used?
  - Overcomes the long message problem
  - Extra security issues to be considered

© Roke Manor Research Limited 2002

56th IETF: ROHC WG Agenda, 2(2)

20:30 - RTP update	
20:30 - Implementer´s Guide	Jonsson (5)
20:35 - Implementation status and test list	Jonsson (3)
20:38 - "IP-only" profile	Jonsson (2)
20:40 - Formal HC notation	Bormann/Ozegovic/Price (60)

ROHC-RTP, DS preparations status, 1(2)

- **MIB & ROHC Terminology and mapping examples**
  ➔ **Submitted to IESG**

- **Implementation status and feature test list updated with first status (incomplete, might even be errors)**

- **IP profile updated (and ready for WGLC?)**

ROHC-RTP, DS preparations status, 2(2)

- Implementer’s guide updated
  - Minor TS encoding clarifications
  - Note about non possible RND flag changes
  - Clarified that CC=0 means that no compressed CSRC list is present, not even an empty list (list header)
  - Clarified which UOR formats are used for profile 2 (UDP)
  - Incorrect reference to the RTP sequence number as 32 bit corrected, was intended to refer to the ESP sequence number
  - Noted that ROHC over PPP (RFC3241) must not negotiate different supported profile sets for IPCP and IPv6CP
  - Document feels rather stable now

56th IETF: ROHC WG Agenda, 2(2)

20:30 - RTP update	
20:30 - Implementer´s Guide	Jonsson (5)
20:35 - Implementation status and test list	Jonsson (3)
20:38 - "IP-only" profile	Jonsson (2)
20:40 - Formal HC notation	Bormann/Ozegovic/Price (60)

ROHC Notation -- why?

Carsten Bormann, 2003-03-17

The 3095 lesson

- Packet formats are non-trivial
  - Overstress RFC box notation
- It is not always clear what goes where
  - Labeling a field in box notation does not mean you know what it **means**
- Interops take much time for debugging the more complex formats

The ROHC notation

- Originally invented for EPIC
  - Has spun off on its own
  - “EPIC” now refers to Hierarchical Huffman
- ROHC-FN Inspirations:
  - BNF
  - ROHC packet classifications
  - Huffman probabilities

ROHC notation: example

```
• ip_header        ::=   ip_version
                         ip_header_length
                         ip_tos ip_length …
• ip_version       ::=   value (4, 4)
• ip_header_length ::=   value (4, 5)
• ip_tos           ::=   static / irregular (6)
                         label (2, ECN)
```

Datagram Congestion Control Protocol Compression profile for ROHC

Julije Ozegovic

FESB Split University of Split

Split, Croatia

E-mail: julije.ozegovic@fesb.hr

Motivation

- DCCP: new transport protocol
  - DCCP WG: active, new drafts released
  - Implementation: planned (before end of summer)
  - Scope: IP/UDP/RTP is mainstream application
  - Compression: considered
- ROHC: in the phase of maturity
  - Profiles: should cover all relevant headers
  - Methodology: formal notation is in development
  - Completeness: all future protocols should be easily described
- DCCP and ROHC: try to define DCCP profile
  - to verify notation completeness
  - to prove formal notation efficacy

DCCP Packet Formats

All DCCP packets begin with a generic header, followed by specific header and options

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |           Dest Port           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Type  |  Res  |                Sequence Number                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data Offset  | # NDP | Cslen |            Checksum           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                  Specific header 0-12 bytes                   /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Options                   /   [padding]   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

DCCP Profile

- Problems: header length
  - Data offset: includes specific header AND options
  - Specific header: length depends on type
  - Options: variable lengths, some implicit (1-byte)
  - Profile: solution found using different option functions
- Profile: not tested on real flows
  - Profile: solution found using different option functions
  - Effort: less than 2 WP (two week persons), incl. DCCP study
- FORMAL NOTATION: conclusion
  - an optimal way for generation of formats
  - ROHC-FN is developing in right direction

Issues in formal notations

- Way too easy to come up with wishy-washy semantics
- Top-down design?
  - Small set of tools
  - Inflexible for new uses
- Bottom-up design?
  - Long way up to application requirements

The ROHC-FN approach

- Start from current (EPIC-evolved) notation
- Define it in terms of well-understood CS concepts
  - Use a (high-level) programming language as the basis
  - Notation instance becomes executable program in this language

What does the program do?

- Not clear. Best case:
  - Compress
  - Decompress
  - Compute some interesting properties of the notation instance (e.g., consistency)
  - Generate code for C implementation
  - Make coffee
  - …

Formal Notation -- Status

- Two attempts at rooting FN in CS concepts
  - Bormann / Prolog / unpublished (Atlanta IETF)
  - Price / Haskell / sent to WG
- Probably useful to maintain both for a while
  - I.e., need not decide immediately
- Use them to think of notation concepts **in a clear, unambiguous way**

What does the program do?

- Given uncompressed header, yields all compressed headers
- Given compressed header, yields uncompressed header
- Enumerate all combinations…

- Cannot (easily) **reason** about itself

ROHC-FN -- Prolog rooting

- Prolog: logic programming
- ROHC-FN: Make use of Prolog Definite Clause Grammars (DCG)
- Describe **relation** between uncompressed and compressed header
- Additional legs for context, bindings

Prolog example

```
• ip_header        -->   ip_version,
                         ip_header_length,
                         ip_tos ip_length …
• ip_version       -->   value (4, 4).
• ip_header_length -->   value (4, 5).
• ip_tos           -->   alter(0.99, static,
                                irregular (6)),
                         label (2, eCN).
```

Prolog rooting: behind the scenes

```
label(N, Label, s(H1,L1,D,V1,P),
   s(H2,L2,D,V2,P)) :-
   takebits(N, H1, [], H2, V2A),
   append(V1, V2A, V2),
   L2 = [l(Label, V2A) | L1].
```

- s(header, labels, discriminator, values, prob)

ROHC-FN -- Haskell rooting

SIEMENS

Roke Manor Research

ROHC-FN in Haskell

Richard Price
Roke Manor Research
(richard.price@roke.co.uk)

© Roke Manor Research Limited 2002

SIEMENS

Roke Manor Research

Overview

- ROHC-FN is currently a brand new language
  - New syntax and new data structures
- Why not reuse an existing language instead?
  - Guarantees that ROHC-FN is well-defined
  - Allows off-the-shelf compilers to be used
- Trick is to find suitable candidates…
  - In theory any sufficiently powerful language is ok
  - In practice we need human-readability too!
  - Need to minimize the “language-specific” overhead

© Roke Manor Research Limited 2002

SIEMENS

ROHC-FN in Haskell

Roke Manor Research

- Haskell is a *functional* programming language
  - Haskell code creates new functions from existing ones
  - ROHC-FN does this for special functions that compress and decompress data (known as “encoding methods”)
- ROHC-FN is already a subset of Haskell (almost!)
  - Only requires minor syntactic changes
  - ROHC-FN code then becomes valid Haskell code

© Roke Manor Research Limited 2002

SIEMENS

Haskell Implementation

Roke Manor Research

- Haskell implementation available from: http://sigcomp.srmr.co.uk/~rjp/RohcFn_Tcp.zip
- What does the implementation contain?
  - Example TCP/IP packet formats (written in ROHC-FN)
  - Library of encoding methods (written in ROHC-FN)
  - Primitive functions (written in raw Haskell)
  - State machine (written in raw Haskell)
  - Test vectors
- No need to implement a compiler for ROHC-FN
  - Any Haskell compiler is also a ROHC-FN compiler

© Roke Manor Research Limited 2002

SIEMENS

Example using IPv4

Roke Manor Research

```
ip_header      =    ip_version.
                    ip_hl.
                    ip_tos.
                    ip_length...
ip_version     =    value 4 4
ip_hl          =    value 4 5
ip_tos         =    static / irregular 8
ip_length      =    inferred_size 16 (-48)
```

© Roke Manor Research Limited 2002

SIEMENS

Example using IPv4

Roke Manor Research

- Suppose we want to compress an IP header
  - Set a mode flag to “compressed”
  - Call the “ip_header” encoding method

```
ip_header       =     ip_version.
                      ip_hl.
                      ip_tos.
                      ip_length...
```

- Calls the “ip_version” encoding method, followed by the “ip_hl” encoding method and so on…

© Roke Manor Research Limited 2002

Example using IPv4

```
ip_version    =   value 4 4
```

- Calls “value” with length 4 and integer value 4

```
ip_hl         =   value 4 5
```

- Calls “value” with length 4 and integer value 5

```
ip_tos        =   static / irregular 8
```

- Calls “/” with a choice of two different encodings
  - “/” calls “static” or “irregular 8” (implementer’s choice)

```
ip_length     =   inferred_size 16 (-48)
```

- Calls “inferred_size” with length 16 and offset -48

© Roke Manor Research Limited 2002

SIEMENS

Example using IPv4

Roke Manor Research

- Eventually Haskell reaches a “primitive” encoding
  - Not defined in terms of simpler encoding methods
- Must provide handwritten code for these
  - Separate code for compression and decompression
  - Mode flag determines which is used
  - Code may include implementation-specific choices

© Roke Manor Research Limited 2002

Issues going forward (1)

- Balance specification/implementation
  - Use oracle functions to include implementation dependent behavior in spec
- Computing the final encoding
  - What does “or” (“/”) do exactly?
  - Completely specified in Notation vs. Additional process (such as EPIC) referenced
  - Body/Discriminator terminology useful?

Issues going forward (2)

- Completeness vs. extensibility
  - Don’t try to be complete this time around
  - Use formal basis as the vehicle for extensibility
- Trade off readability and formal roots
  - Notation must remain as accessible as RFC box notation
- Context management vs ROHC-FN
  - How does FN interact with the context?
  - Maintain “context families” in the notation?
    - Truncation of the family is the fun part

Where does this discussion go?

- Scope section in ROHC-FN document:
  - Define requirements for ROHC-FN
  - List non-requirements as well