

Layered Syslog Architecture

Rainer Gerhards

Adiscon

rgerhards@hq.adiscon.com

2003-11-26

Current Status

- RFC 3164, 3195, syslog-sign and -international each specify message format, transport specifics and on top of that some specific functionality.
- Each RFC/ID makes slight changes to the format, so there are minor inconsistencies.
- This makes it hard to
 - support all standards in a single program
 - e.g. transport a syslog-sign message over a 3195 channel
 - build relay chains with different RFCs “in them”

A Solution

- Experience in Protocol Design tells us that forming multiple layers, each one with defined “interface” to its lower any upper layers as well as a defined functionality helps to keep things
 - simple
 - extensible & powerful (new functionality “immediately” available to older parts)
 - “change-ignorant” (new functionality does not break existing code)
- Many successful protocols use this approach (e.g. SMTP, BEEP)

Syslog Protocol Layers

“Applications”, e. g. Signatures, International Characters

Message Format & Syslog Semantics

Transport Mappings

Transport Layer/Mapping

- Mapping to UDP – currently, transport parts of RFC 3164 can serve as this
- Mapping to TCP – currently, transport parts of RFC 3195 can serve as this
- Potential for other mappings
 - SNMP (“syslogish” messages are send via SNMP TRAPS by some devices)
 - “unreliable”, raw TCP for those who really need it [I don't say it is a good idea, though ;-)]

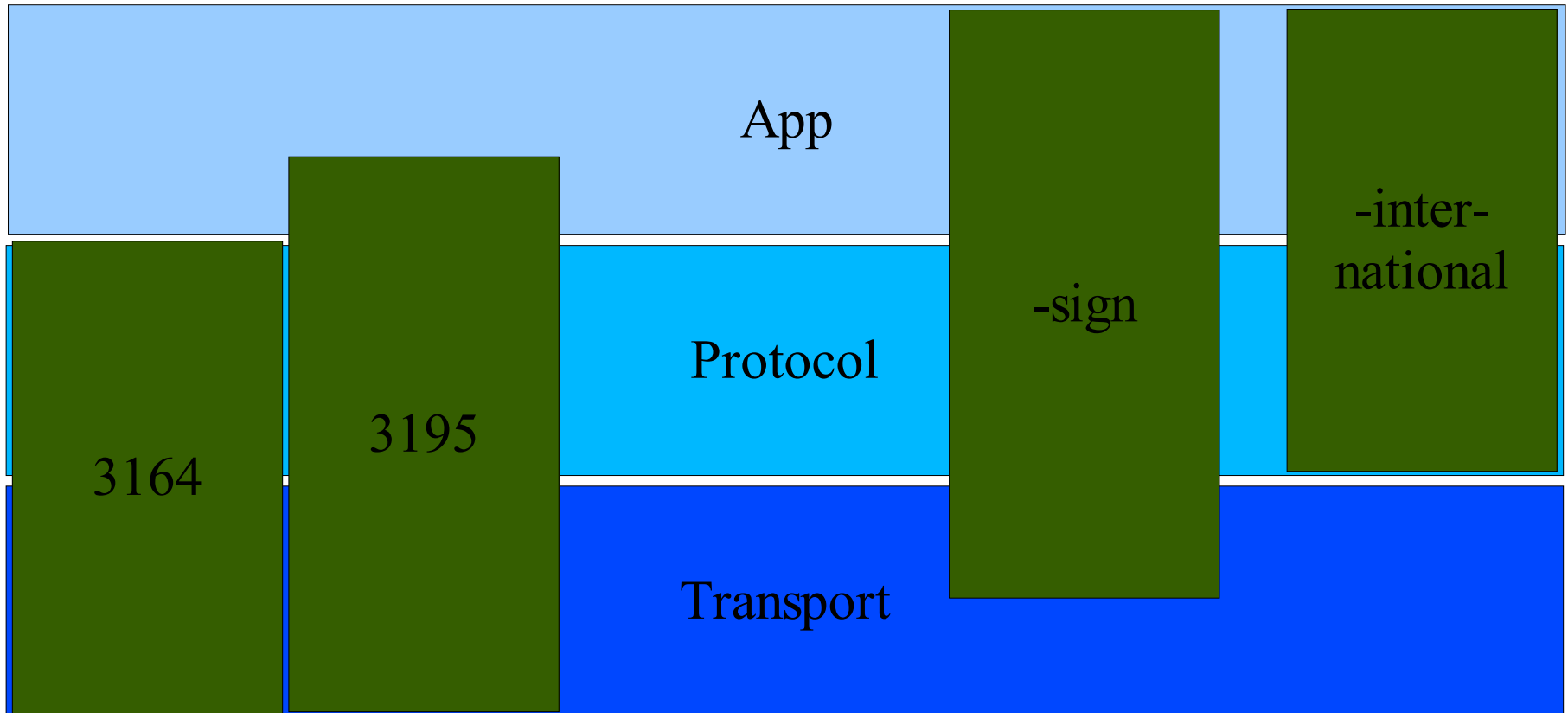
Formats & Semantics

- Syslog communication model
 - Simplex (UDP!)
 - Duplex (RFC 3195)
- Device Roles
- Message Format (frame?)
- Cookies

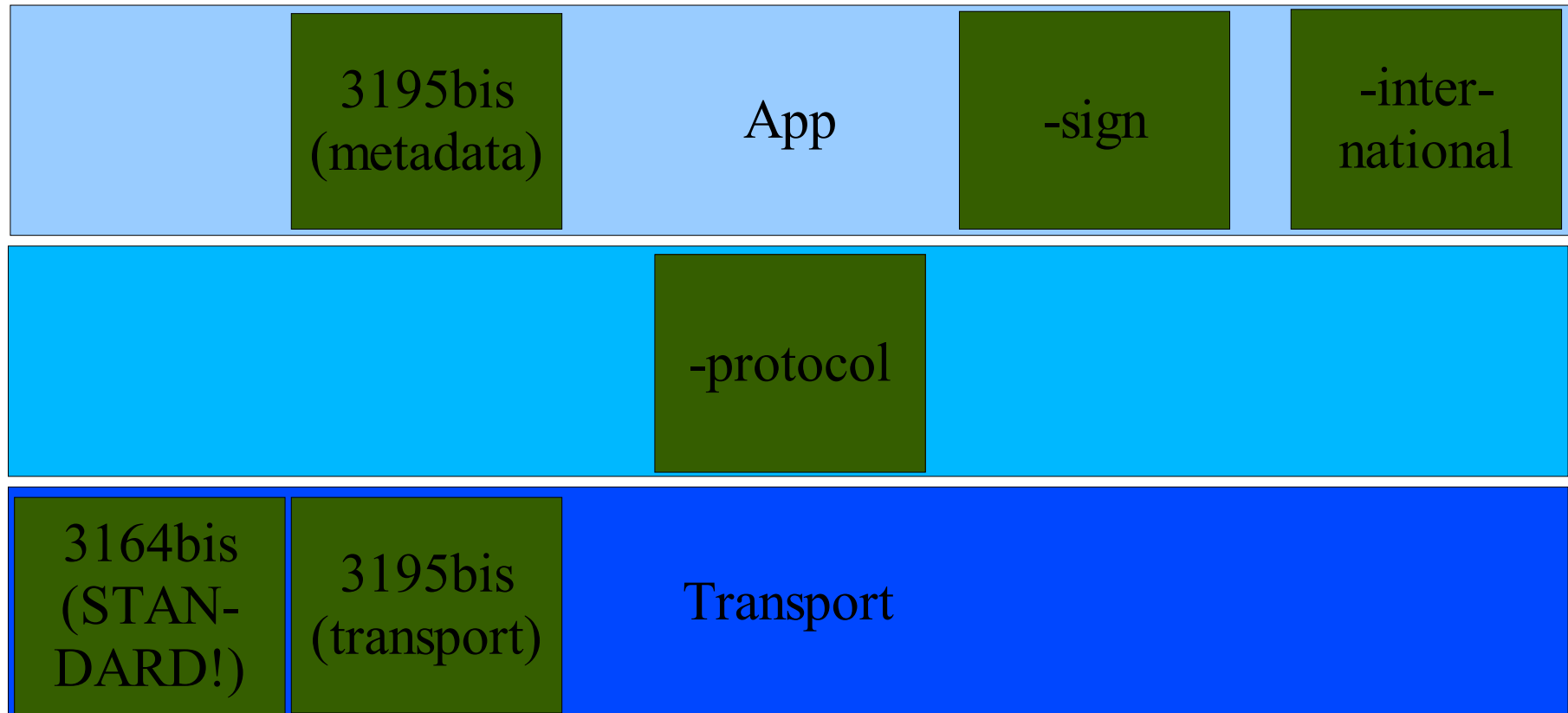
Applications

- Digital Signatures
- International Character Sets
- Metadata? [like those exchanged in 3195/COOKED]
- ... more to come? [at least we are prepared for it]
 - I personally could think – at a later stage – about some ID recommending a content format. This is outside of the scope of the current WG, but one of the major frustrations with syslog in the real world.

Current RFCs/IDs



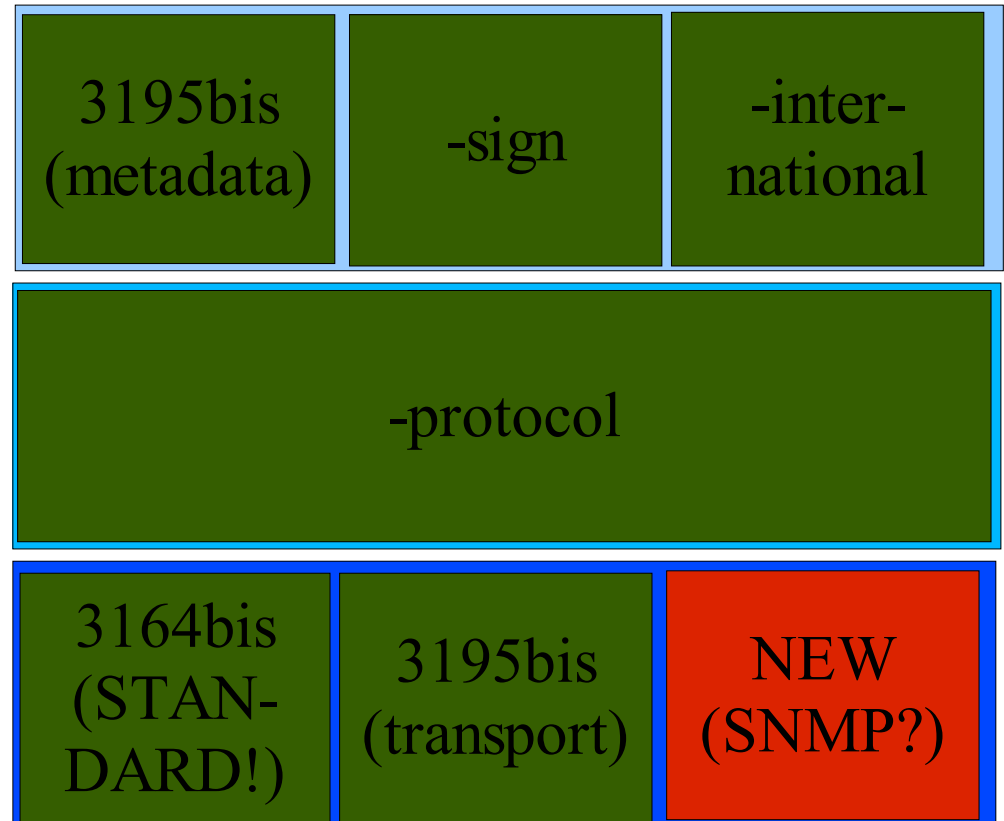
What Could be Done...



This would mean 6 RFCs instead of 4, but each one shorter and interoperating. Thanks to consistency, they would hopefully be easier to implement and manage (operations management).

What happens if Something New is added?

- Everything else would continue to work
- The new functionality would be available to all existing syslog “parts”



What is in -protocol?

- Syslog “system” components
 - Protocol layers described
 - devices, relays, collectors
 - simplex communication structure (is duplex an option at all...?)
- Message format
 - common header
 - non-structured, 8 bit message part
- Cookies (concept & some defined)
- General security considerations

Is there much to do to create -protocol?

- No, most parts can be simply taken from the already existing RFCs and IDs
- Things to actually discuss:
 - Simplex structure
 - Inclusion of version number in header?
 - Include fragmentation in header or cookie?
 - Cookie format
 - Other RFCs and IDs can also quickly be adapted to -protocol.

Does anything need to be Broken?

- Thankfully, no existing code needs to be broken.
- There is one topic regarding the colon in TAG which could cause some (minor) issues ... however, this is “inherited” from syslog-sign & -international, so it is not a new issue.

Communications Model - Status

- Traditionally, syslog is simplex, that is a message is send and the sender does not have **any** idea if the message will ever be received.
- RFC 3195 introduces some (limited) duplex capabilities. So a sender could at least know if the (next) receiver was able to receive. But: that did not mean the message arrived at the ultimate destination!

Communications Model – What to do?

- We can specify syslog
 - to be simplex communication in general
 - optionally allow a transport mapping to acknowledge next hop delivery.
- We could try to do more (probably a bad idea)
- If we would like to optionally allow more than **next hop** acknowledgement, we would need to think very hard about this – it has a lot of implications...

Cookies

- Used to implement “extended functionality”.
- First introduced in syslog-sign
- Can be expressed in an
 - XML-like form
 - or as a sequence of special characters
- Both forms very unlikely to be seen in already existing syslog data (extremely low error probability do to cookie miss-detection)

XLM-like Cookies - Format

- `<cookie-name param="value" />`
- XML **subset**, so that no XML parser is necessary to process message:
 - No containers allowed??? (think twice before posting this!)
 - Parameters can be optional, but must be in a specific sequence
 - Unknown parameters must be ignored (no error)
- Easily extensible by adding new parameters

XLM-like Cookies – Pro & Con

- Con: Could be confused with real XML and thus parameters be written in wrong sequence
- Con: Look like more work to implementors (and could cause some hesitation to pick them up)
- Con: a bit lengthy (but is this really an issue?)
- Pro: easy to process when an XML parser is already available (3195!)
- Pro: Easier to read for humans
- Pro: extensible, so no versioning issues

Message Format Version

- A format version number in the header can be a life-save when there is need to upgrade the format (for a good reason ;)) in the (distant?) future.
- An integer incremented each time the format changes (IANA?)
- Could be optional and – when missing – set to be “this format” (1), so all existing code would already do it right.
- Making it optional in the header is the tricky part (without a cookie...)

Fragmentation

(aka “Oversize Messages”)

- Fragmentation could be done in a cookie or in an header extension
- A cookie may complicate things, as this cookie is extremely vital to all operations (EVERY receiver MUST recognize it, or things can go wrong)
- If done in the header, we have an interop-issue with older code.

Probable Solution?

- Use a single optional cookie for both the message format AND the fragmentation
- Make the fragmentation part in the cookie optional
- Not extremely elegant, but solves all interop issues with existing code
- Cookie must not be written if we use current format AND no fragmentation BUT it is recommended to write the version in any case.