Shim6 protocol

draft-ietf-shim6-proto-02.txt

Erik Nordmark
erik.nordmark@sun.com

Overview

- What isn't new
- Brief introduction to protocol
- Open Issues

Unchanged direction

- Shim between IP endpoint sub-layer and IP routing sub-layer
  - Below fragmentation/reassembly, below ESP/AH
- ULID – an IPv6 address
  - Selected by getaddrinfo(), application, or transport protocol just as today
- HBA/CGA for security
- Deferred context establishment
- Host-pair context state
  - NOT per TCP connection

Protocol Overview

- Packet formats
- 4-way context establishment exchange
- Context taredown
- Context recovery
- Sending ULP messages
- Receiving ULP messages

Shim message formats

- Allocate a new IP protocol number to Shim6
- All have 16 bits in common

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  |       0       |P|             ...             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               .....                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Fields:

Next Header:   The payload which follows this header.

Hdr Ext Len:   In units of 8 octets, excluding first 8 octets

P:              Set to 1 for payload messages, and 0 for control messages
```

Shim control messages

- A terminal header – just like ICMPv6
- Base header

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  nxthdr = 59  |  Hdr Ext Len  |0|     Type    |Type specific|0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Checksum             |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                    Type specific format                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
`

Payload message

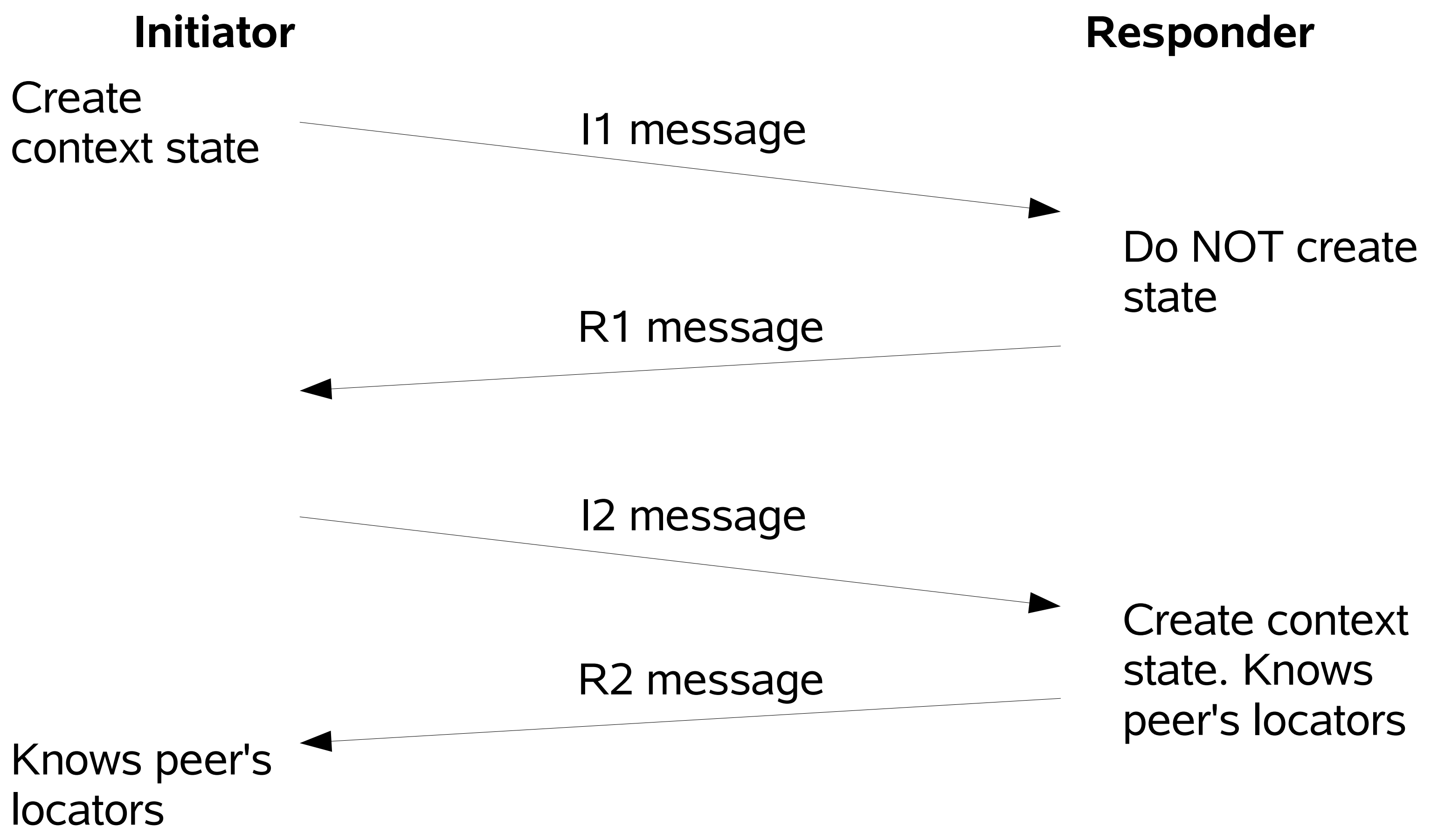
- Have an 8-octet extension header (the shim6 payload message) carry the next header value and the context tag

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  |       0       |1|           Reserved          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Context tag (continued)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Fields:

Next Header:   The payload which follows this header.

Hdr Ext Len:   0 (since the header is 8 octets).

Context tag:   32 bits
```

Deferred Context Establishment

- Hosts start doing ULP traffic as today
  - E.g., TCP packets flow
- Hosts use some heuristic to decide when to setup context state
  - For instance, after sending or receiving 50 packets for ULID pair
- Could have some APIs to affect this
  - Such as IPV6_DONTSHIM socket option to make the context not be created/used

Context establishment



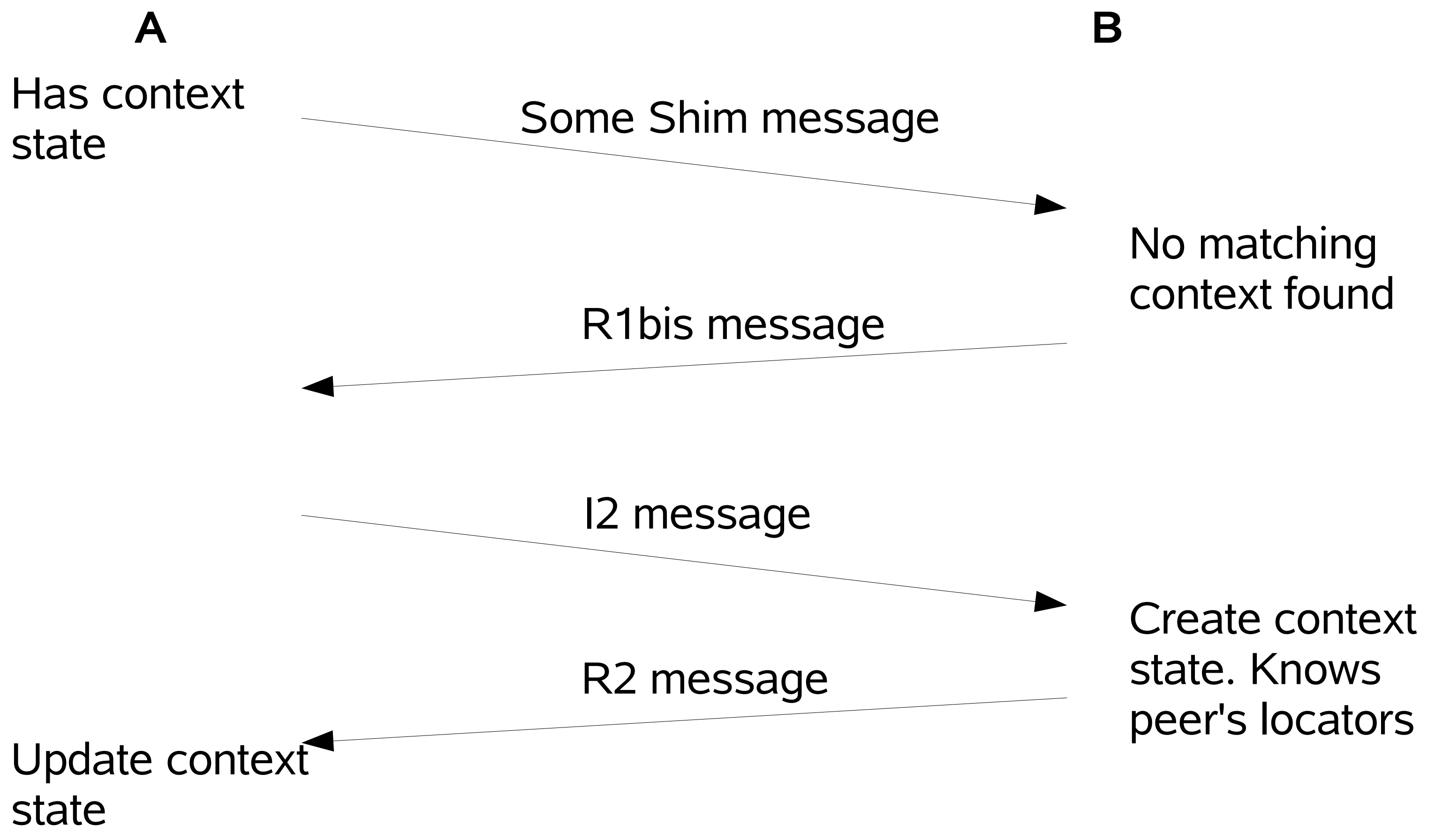
Context taredown

- Uncoordinated state removal
  - Each host can remove state at any time
- Undesirable to remove context state when it is in use by ULPs
  - But implementations can't always tell, e.g., UDP “sessions” in application space
- Question: can one operate in asymmetric mode where the “server” garbage collects the context state even thought it is used?
  - Can't always recover from failures in this case

Context recovery

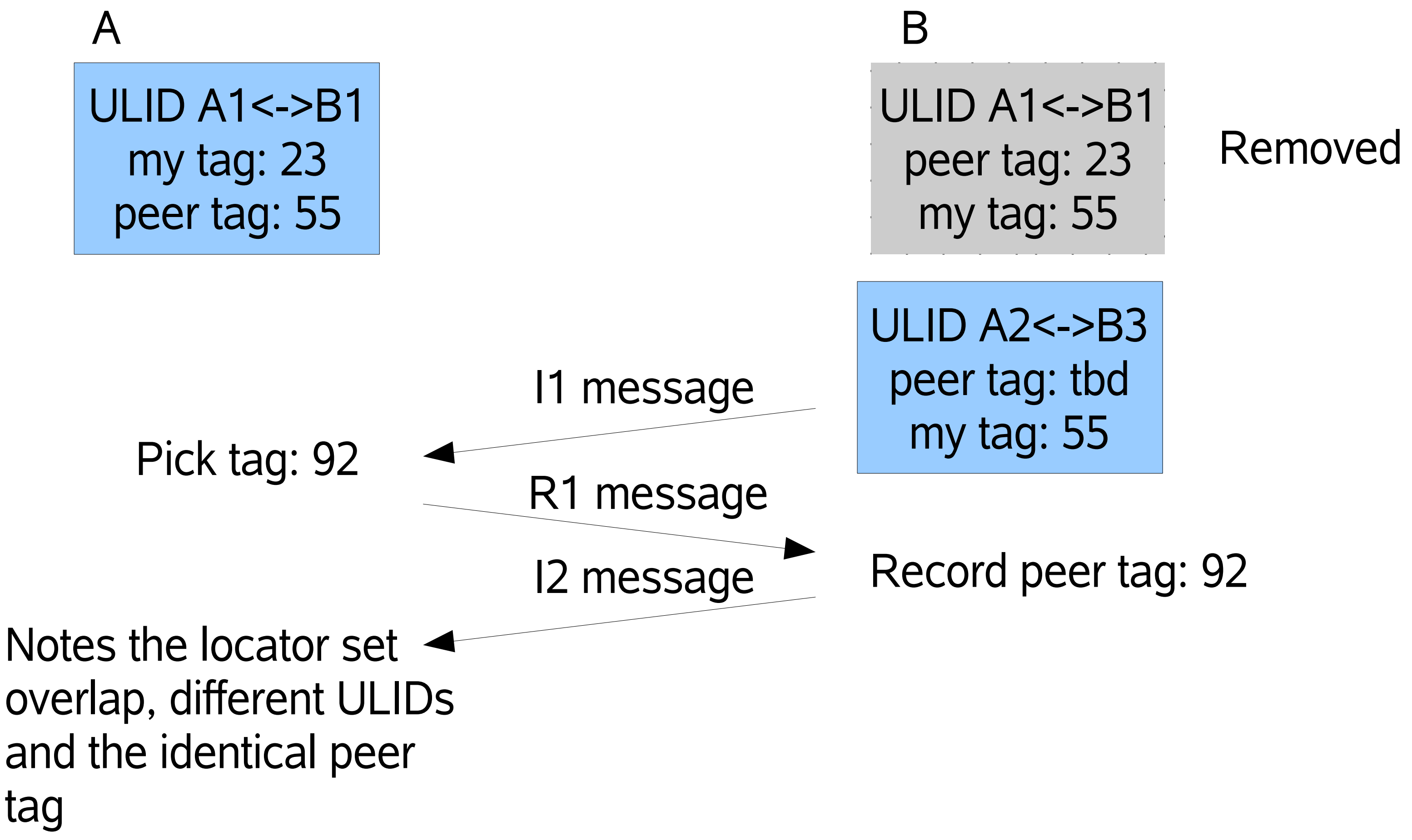
- Should the peer have removed context state while it is in use
  - Garbage collected too early, or crashed
- Recovery triggered by the peer receiving some shim message (control or payload)
- Re-uses context establishment model to re-create the context state
  - With some modifications due to difference in information in I1 message and “triggering” shim6 message
  - See R1bis message from Marcelo

Context recovery (2)



Context “confusion”

- Due to uncoordinated taredown, might reuse a context tag that the other end views as in use



Sending ULP packets

- ULP passes down message to shim
- Shim looks up ULID pair
  - Context not found
    - Send unmodified ULP packet
    - (Track #packets for deferred context establishment)
  - Context found, and current locator pair = ULID pair
    - Send unmodified packet
  - Context found, locator pair != ULID pair
    - Add 8 byte shim6 extension header to packet
    - Change IPv6 source and destination to be the current locator pair

Receiving packets

- Stack demultiplexes based on the next header values
- If shim6 next header found
  - Check if payload bit is not set
    - Then demultiplex on shim6 message type
  - For payload messages
    - Use context tag etc to find context state
    - Not found, then send R1bis message
    - Found: replace IPv6 source, dest with ULIDs before passing packet to ULP

Option Formats

- Each option a multiple of 8 octets
- Same format as in HIP
  - Makes it easier if we later want to extend shim6 with HIP features

```
  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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type            |C|            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
/                          Contents                             /
/                                               +-+-+-+-+-+-+-+-+
|                                               |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option Name	Type
Validator	1
Locator List	2
Locator Preferences	3
CGA Parameter Data Structure	4
CGA Signature	5
ULID Pair	6
Packet In Error	7
SHIM6 Event Option	8

Open Issues

- A few in the document
  - Forking
  - Context tag reuse
  - Renumbering considerations
  - Terminology
  - Timers
  - Size of context tag
  - Packet injection, context tag lookup
  - R2 being lost
  - Asymmetric context state storage
  - When to verify locators

Forking the context state

- Different ULP streams can be sent using different locator pairs for the same ULID pair.
  - No protocol extensions are needed if any forking is done independently by each endpoint.
  - Being able to have A tell B which ULP packets to send over which fork is both complex, and interacts poorly with the ULP on B also being able to decide.
  - Need API and ULP mechanism to use forked context. [Not in this document]
- Forking implies that reachability detection must be applied separately for each locator pair that is in use [Not in this document]

Context tag reuse?

- What happens when a host runs out of N bit context tags?
- When is it safe for a host to reuse a context tag? With the unilateral taredown one end might discard the context state long before the other end.
  - The "context confusion" TBD will presumably take care of this
  - Check this and move forward

Renumbering Considerations

- Should a host explicitly fail communication when a ULID becomes invalid (based on RFC 2462 lifetimes or DHCPv6), or should we let the communication continue using the invalidated ULID (it can certainly work since other locators will be used).
- Leave for further study

Terminology

- Should we rename "host-pair context" to be "ULID-pair context"?
  - If we've decided this is per ULID pair that might make sense.
- Yes, change this

Timers

- We need to pick some initial retransmit timers for I1 and I2.
  - Is 4 seconds OK?
  - Yes
- Should we require that the R1 verifier be usable for some minimum time?
  - So that the initiator knows for how long time it can safely retransmit I2 before it needs to go back to sending I1 again?
  - Pick 10 seconds

Size of context tags

- Should we expand the context tag from 32 to 47 bits?
  - Doesn't imply any larger packets
  - Change to 47 bits

Packet injection; context state lookup

- Should we make the receiver not use the source locator to find the context, but instead only use the context tag? (and optionally, the destination locator)
  - This would provide some flexibility for the future. The potential downside, which we would need to understand, is packet injection. *If* there is ingress filtering, then we get some extra checking by including the source locator in the lookup. But an on-path attacker can inject packets at will, whether the source locator is part of the lookup or not. An off-path attacker would have a hard time to guess a 47-bit number.

R2 being lost

- Include locator list in R1 message to deal with R2 being dropped?
  - Would seem to require including HBA/CGA information.
  - But CGA signature needs to be “live” i.e. include some timestamp or nonce, to prevent replays
    - This would imply generating a PK signature in order to respond to an I1. Bad
- Or assume that I2 (or I1) will be retransmitted
  - And payload extension headers sent from responder to initiator before R2 has been received, will be dropped. [Add wording if needed to say this]

Asymmetric context state

- Should we allow a host to intentionally discard the context state, with the assumption that the peer is responsible to maintain it, and detect failures?
  - This might be useful in asymmetric case, e.g. a server which serves lots of clients
  - Can't recover from all failures.
    - If the client doesn't send anything for a while, and when the server starts to send the locator pair doesn't work any more. Server can do nothing since it doesn't have a context with alternate locators, and the client can't possibly know that the server might be having problems reaching it.
  - Make into "for further study"

When verify locators?

- When does a host need to verify the locator list?
  - Immediately i.e. before accepting packets from those locators as the source address?
  - Or before sending packets to those locators?
  - There are some issues if it isn't verified immediately
    - Would allow an on-path attacker to send bogus update messages which can not be verified; that would potentially make the host no longer accept packets from the actual locator that the peer is using, and when it tries to verify the locators it would find that they are "bad" and has no alternate peer locator it can use.
- Alternative is to not use source locators to find context (see packet injection issue)

TBDs in the document (1)

- Add descriptions
  - forking
  - API to turn it off
  - updating locator pairs
  - rehoming
  - initial contact
  - security considerations
  - IANA rules for shim6 message types and option types

TBDs in the document (2)

- Make consistent with failure/reachability detection draft
  - In which draft do the the packet formats go?
  - Reserve message types and option types. Rest in separate draft.
- Work out context recovery – R1bis message
  - including when ULID pair != locator pair
- Work out context confusion
  - accept new context and recreate context for other ULID pair

Next Steps

- Fill in the TBDs with text
- Close the easy open issues
  - Provide input if you care about particular issues
  - Locator verification/packet injection issue requires some thinking
- Remaining open issues will be turned into “for further study”
  - Basically things that we don't want to forget as we learn from implementation experience
- Get a complete (implementable spec) in a few weeks