# RELOAD Base Update

draft-ietf-p2psip-base-13 IETF 80

Dr. Cullen Jennings, Ph.D.

Distinguished Engineer, Cisco VTG

fluffy@cisco.com

# DRR

- Specified in a separate draft
  - Need minimal support in base draft for forward compatibility
- Proposal
  - Exception in § 4.1.2 to allow extensions to adjust state keeping rules:

    > Whatever algorithm is used, unless a FORWARD_CRITICAL forwarding option or overlay configuration option explicitly indicates this state is not needed, the state MUST be maintained for at least the value of the overlay reliability timer (3 seconds) and MAY be kept longer. Future extension, such as [I-D.jiang-p2psip-relay], may define mechanisms for determining when this state does not need to be retained.

  - `IGNORE-STATE-KEEPING` forwarding extension in `draft-jiang-p2psip-relay-05` to use via list
  - Agreement from Jennings, Lowekamp, Even, Rescorla

# Configuration Refresh

- Previously, just expiration time
- Now: retrieve configuration at a randomly selected time in the future

# Configuration Defaults

- Many settings in which there were no defaults now have them
  - Topology plugin (`CHORD_RELOAD`)
  - Port (6084)
  - Clients-permitted (`chord-reactive = true`)
  - ...

# Resolving Simultaneous Connects

- What happens if two nodes try to simultaneously connect to each other
  - End up with two connections
- Tie-breaker algorithm [Chen]
  - Comparison of Node-Id $A$, $B$ as unsigned integers
  - Keep $A \rightarrow B$ connection iff $A > B$
  - Larger node-id sends a `Error_In_Progress` error (new)

# Replica/Topology Shift Stores and Lifetime

- At time $t_0$ data is stored at $A$ with lifetime $T$
- At time $t_1$, $A$ stores to $B$ and needs to send lifetime $T'$
  - Draft didn't define the adjustment algorithm
- § 6.4.1.1 now has an explicit algorithm:
  - $T' = T - (t_1 - t_0)$

# Triggered ConfigUpdates

- $A$ tries to store item of kind $k$ to $B$ but $B$ doesn't know about $k$
  - $B$ sends back `Error_Unknown_Kind`
- This is a signal that $B$'s config is likely out of date
- New requirement (§ 6.4.1.2) for $A$ to generate a `ConfigUpdate`

# Resource-ID Computation Algorithm

- Used to be in the configuration document but now it's defined by the overlay algorithm
  - Proposal: SHA-1 for Chord
- Bug in the current document: § 9.2 specifies truncation to 128 but node-ids vary between 128 and 160 bits (§ 5.3.1.1)
  - This is obviously bad
  - Proposal: SHA-1 truncated to node-id length

# Clarify Congestion Control Considerations

- Wasn't totally clear what the requirements were for flow control algorithms. Reworded in § 5.6.3.1

  "Because the receiver's role is limited to providing packet acknowledgements, a wide variety of congestion control algorithms can be implemented on the sender side while using the same basic wire protocol. In general, senders MAY implement any rate control scheme of their choice, provided that it is REQUIRED to be no more aggressive then TFRC[RFC5348].

  The following section describes a simple, inefficient, scheme that complys with this requirement. Another alternative would be TFRC-SP [RFC4828] and use the received bitmask to allow the sender to compute packet loss event rates."

# Extensive Uncontroversial Technical and Editorial Changes

Special thanks to Marc Petit-Hugenin for an amazingly thorough reviews.

# Open Issue: How to get a certificate with multiple Node-Ids?

- Nothing in the draft here.
- Consensus seems to be `POST` argument
- Concrete Proposal: `nodeids=X` URL parameter

# Open Issue: Signing With Multiple Node-Ids

- Signatures are currently tied to certificates (`SignerIdentity`)
  - But what about certificates with multiple Node-Ids?
- Proposal: Add a Node-Id indicator to signature

# Proposed Multiple Signature Syntax [Petit-Hugenin]

```
enum { reservedSignerIdentity(0),
        cert_hash(1), cert_hash_node_id(2) (255)} SignerIdentityType;

struct {
   select (identity_type) {
     case cert_hash:
       HashAlgorithm      hash_alg;              // From TLS
       opaque             certificate_hash<0..2^8-1>;

     case cert_hash_node_id:
       HashAlgorithm      hash_alg;              // From TLS
       opaque             certificate_hash<0..2^8-1>;
       NodeId             node_id;

     /* This structure may be extended with new types if necessary*/
   };
 } SignerIdentityValue;
```

# Attach for Initial Connections

- Ambiguity about whether initial (non-bootstrap) connections require `Attach`
  - This includes client connections
  - This allows determination of Node-Id
- Now clarified in § 11.

# Open Issue: Who Signs Stores?

- § 6.4.1.1 (`StoreReq` processing) requires verifying the stored value signatures
- § 6.3.x (access control policies) is phrased in terms of the signed request
- Both signatures matter, but this is wrong
- Proposal:
  - Check stored value signatures against access control policies (§ 6.3.1)
  - Check the request signature against either access control policies (original store) or plausible responsible node (replica store)

# Open Issue: Refreshing Finger Table

http://www.ietf.org/mail-archive/web/p2psip/current/msg05564.html