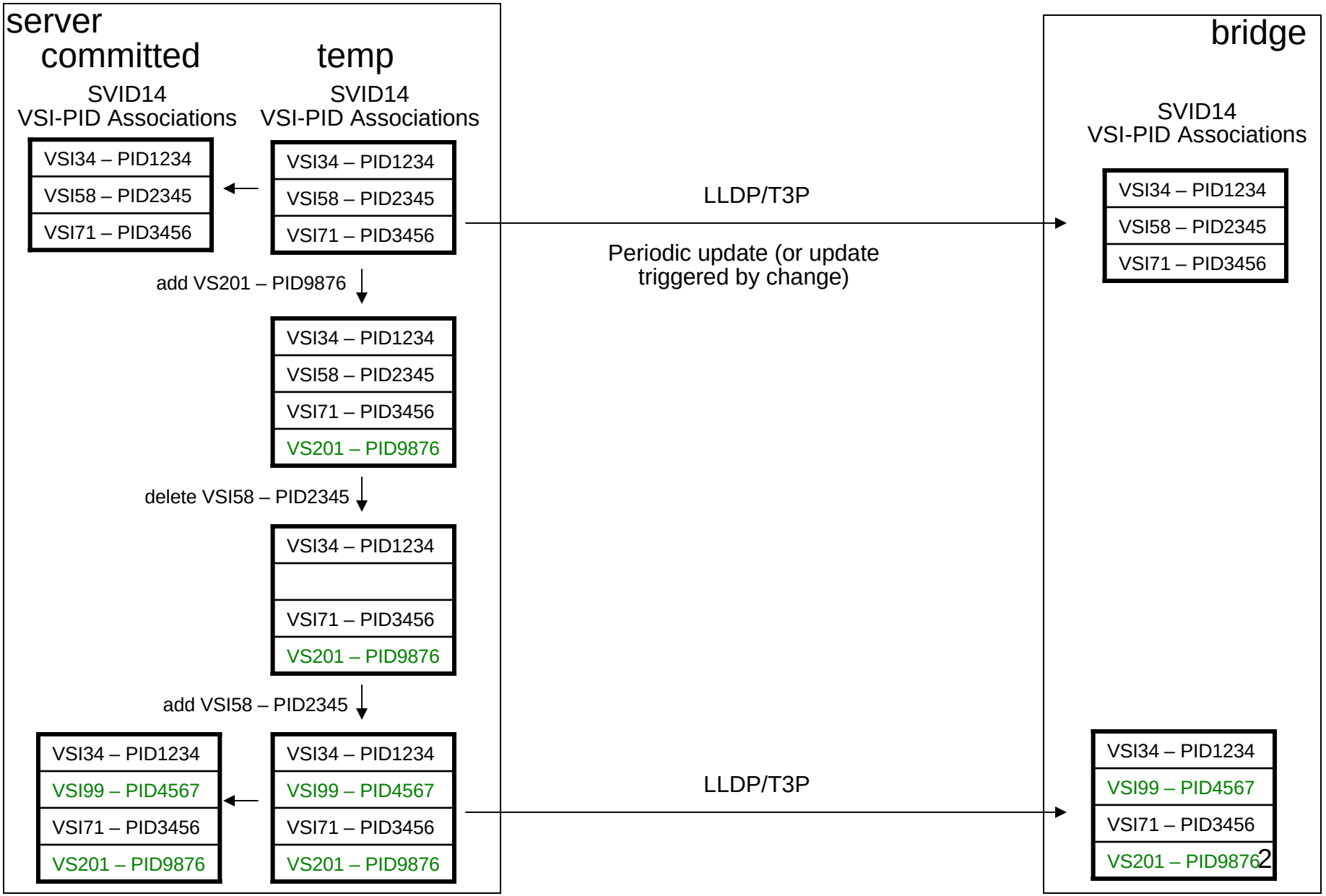
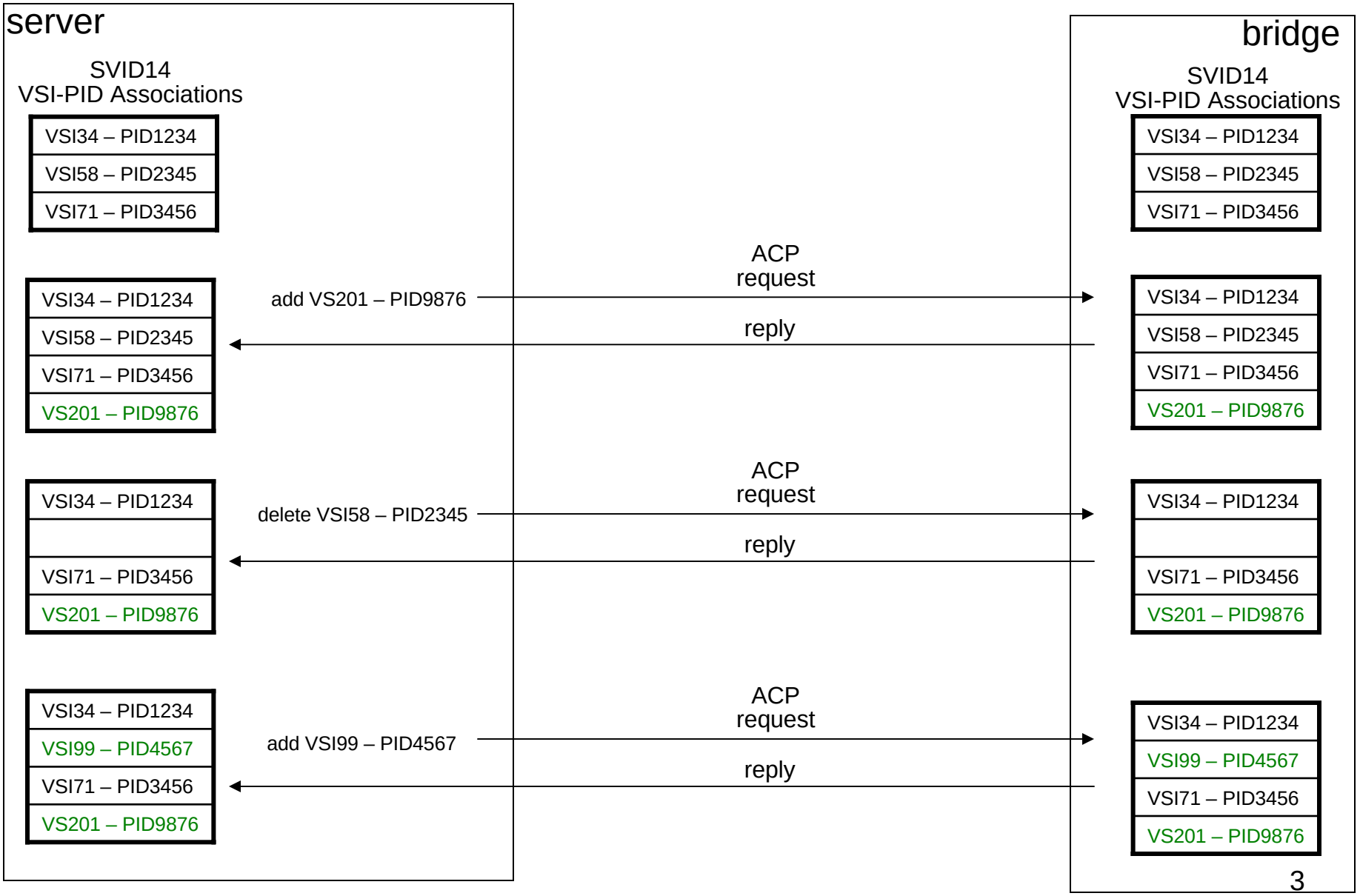
Comparing LLDP/T3P and ACP for use in evb protocols

Bob Sultan (bsultan@huawei.com)
Ben Mack-Crane (tmackcrane@huawei.com)

LLDP/T3P Model



Database *Update* Model



Questions/comments in comparing the models

- It has been suggested that a benefit of using LLDP is that this protocol is *designed* to ‘synchronize’ the database of the LLDP receiver with that of the LLDP sender;
- The function that LLDP provides is the delivery of the database from the sender to the receiver;
- The receiver then has two copies of the database that can be compared for discrepancies;
- Discrepancies can be reported to the operator;
- LLDP provides *no* method to synchronize or harmonize the sender and receiver versions of the database;
- If we assume that the sender (server) has the ‘valid’ version of the database, ‘synchronization’ could be achieved by enforcing a rule that the receiver (bridge) accepts the sender’s version of the database as authoritative;
- So, if it is important that ‘all records in the database be committed’ or ‘no records in the database be committed’ at the receiver then LLDP is a useful way to achieve this;

Questions/comments in comparing the models

- So, is this ‘atomic’ commitment of the database a requirement for evb?
- The argument put forth is that the VSI bindings associated with a particular vPort should be committed as a unit as they are interdependent with respect to their use of resources like vPort bandwidth;
- That is, some outside agent has computed the appropriate set of bandwidth reservations for VSIs associated with each vPort; these are intended to be committed as a unit;
- But this is inconsistent with our evb model that the bridge evaluate each individual binding (DB record) and reject those for which it does not have resources;
- So, this is *not* a case where we guarantee that the sender’s proposed version of the database is accepted or rejected by the receiver;
- That is, there is no requirement to ‘synchronize’ databases;

Questions/comments in comparing the models

- We need only ‘synchronize’ individual database records; that is, the sender proposes that a record be committed by the receiver, and if that record is committed by the receiver, a reply is sent to the sender indicating that it can commit the record;
- It is not at all clear that LLDP is well-suited for this application; it would seem that a simple request/reply protocol is a better fit;
- But, a request/reply protocol has problems; for example, what happens if the server reboots after a record has been committed by the bridge?
- Committed records are associated with a TTL; on TTL expiry, the record is aged-out;
- The sender can repeatedly request commitment of the same record; the only impact is that the timeout value is reset (ie. idempotency);
- If the sender disappears, the record will age-out;

Questions/comments in comparing the models

- If the sender re-appears, it makes a new commitment request which is evaluated independent of whether or not an earlier commitment has, or has not expired;
- It might be argued that, although evb does not require database synchronization, LLDP is still a good fit for the application because it allows multiple records to be sent in a single frame;
- However, the current proposal for ACP allows a fully equivalent level of ‘packing’;
- It might be argued that LLDP is useful even when the receiver is allowed to commit some records and not others;
- For example, after receiving a set of records for proposed commitment, the bridge could evaluate each records to determine whether that record can be committed; each such record is associated with a state (committed, or uncommitted); the bridge then sends an LLDP message back to the server with the state of each record; the server then accepts the states proposed by the bridge;

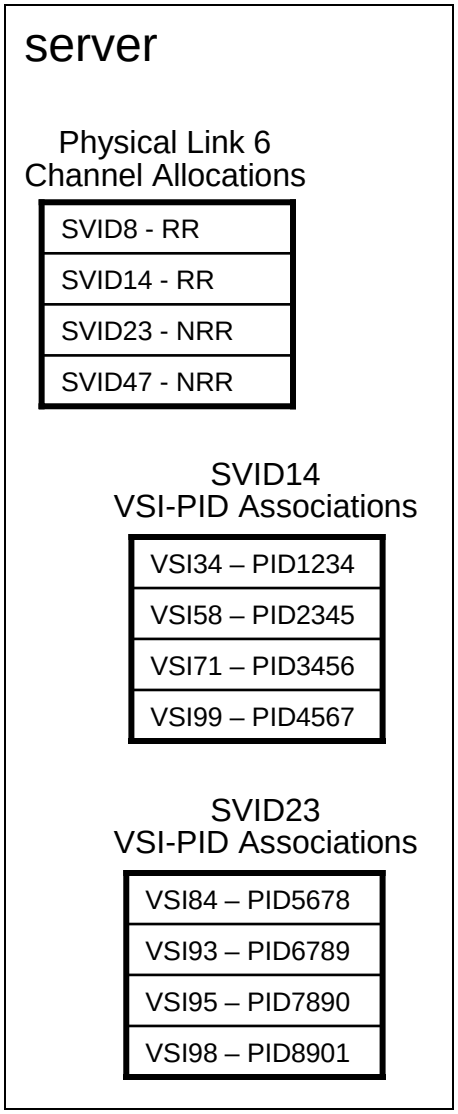
Questions/comments in comparing the models

- It might be argued that the LLDP message from the bridge to the server *is* a ‘reply’; its content depends upon the LLDP message previously sent from server to bridge;
- If we overlook this issue for a moment, we have another problem; the server installs a record with key X in its database; some time later, the server receives an LLDP message from the bridge containing a record with key X; how does the server know whether
  - A) the record X in the server database is the same record X for which the ‘reply’ has been received; in this case the record X in the server database should be replaced by the record X in the reply;
  - B) the record X in the server database has not yet been sent to the bridge in an LLDP message; it should not be replace by the record X in the reply;

Summary

- LLDP is designed to advertise a database and to allow neighbors to identify discrepancies between local and remote versions of the database;
- It is a simple use of LLDP to install the received version of the database as the authoritative version;
- LLDP is *not* designed to selectively commit *individual* records within the database;
- A request/reply protocol with timeout is a better fit for this requirement;
- On its face, it would appear that a protocol that advertises an entire database to update a record will either (a) require a great deal of bandwidth or (b) introduce significant delay in updating records if the frequency of advertisements is reduced;
- This existance of LLDP is certainly a point in its favor, but we must evaluate whether the changes to LLDP needed to support this application will require more effort than the introduction of an additional protocol;

Additional question about the LLDP/T3P Model



- The database on the server is really hierarchical as shown on the left;
- In the LLDP/T3P model, what exactly is the ‘database’ that is advertised?
- Is it everything at the left (in a single LLDP message); Does each table form an individual database?
- What criterion is used to determine the scope of databases?
- If the criterion is inter-dependency, then why would we assume that there is more inter-dependency between records on one vPort than there would be on records associated with different vPorts?
- This is unclear in the current LLDP/T3P proposal.