Schema Replication

Some Requirements

- Schema Service must be ‘globally’ consistent
- Schema Service must continue under duress

Some Use Cases

- Schema Service receives a Producer table definition which already exists
    - Schema Service receives request to add a table definition. Table definition already exists within the Schema Service. No replication carried out.
- Schema Service receives a new Producer table definition
    - Schema Service receives request to add a new table definition. Table definition is then copied to each Schema Service. Each Schema Service returns an acknowledgement. Schema Service receives all replies and then returns.

Topologies

- Centralised
  - Based on Master – Slave relationship
    - Master Schema writes new table definitions to slaves
    - Next ‘best’ Schema Service used if failure occurs
- Distributed
  - Based on Registry Replication model
    - Each user of the Schema API uses the same Schema Service
    - Each Schema Service synchronizes new table definitions with neighbouring Schema Services

Centralised Approach

- Pros
  - Naturally fits-in with the global view of the Schema
- Cons
  - How do we ensure all users of the Schema API use the same global Master Schema?
    - Difficult to enforce during an R-GMA restart
  - Re-election requires lots of co-ordination
    - Difficult to implement with potential caveats

Distributed Approach

- Pros
  - Simpler to implement
    - Avoids complicated re-election algorithm
    - Code re-use - Registry Replication
    - User of the Schema API picks any Schema Service without having to work out which is the Master
- Cons
  - New tables must be synchronized with all Schema Services
    - Problem also applies to the Centralised approach

Some Implementation Ideas

- Republisher wont work
  - Cant easily synchronize data
  - Difficult testing and debugging
  - Dont want to change the Republisher implementation in case it knackers the Replication
    - Probably wont know its broken until deployed on a test bed with system testing switched on

Some Implementation Ideas cont …

- JAXB for fun and profit
  - You provide the XML Schema and run a Compiler that generates a customized java parser. Parser reads in XML docs conforming to the Schema and then provides a nice memory struct.
  - Possible to convert the XML directly into a DB (using the Apache jaxme)
- Will simplify both Registry and Schema Replication

Summary

- Distributed approach is easier to implement
  - Fits in nicely with Registry Replication algorithm
  - Potential for code reuse
- Challenging issues with either approach
  - Have to ensure all *back-up* Schema Services are consistent