draft-prz-lsr-hierarchical-snps-00 Optimizing CSNP Load at Size

Tony P, HPE of former Juniper

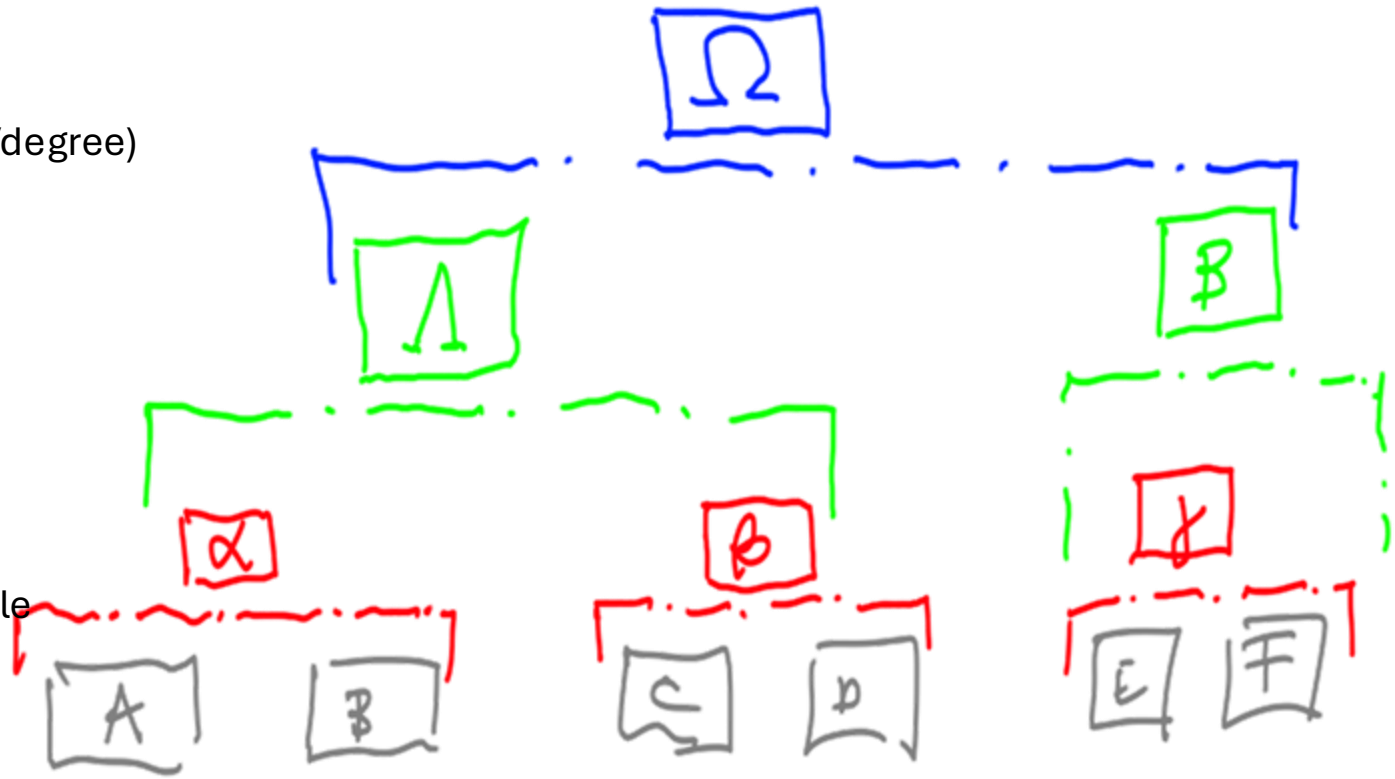
First discussion framework

What's the Problem?

- CNSPs at massive LSDB size and lots adjacencies cost
  - Computing
  - I/O
- On stable network with correctly working flooding basically 100% of that periodic load is useless overhead
  - Well, it's actually useful (very) if flooding hiccs up ;-)
  - Caching helps but not with I/O and it's tricky to do when CSNP ranges within packets shift (re-packing)

Recycle of old idea with ISIS specific twists

- Merkle Trees published ‘87
- Tree of hashes subsuming lower rank (level/degree) hashes
- We call the bottom rank 0 for now
- Rank 0 are basically full CSNP entries A--F
- Rank 1 α-ɣ will be hash over a set of CSNPs
  - XOR’ing of a good XOR hash of CNSP entry (fragment-id, csum, seq#)
- Rank 2 Ά—B will be hash over set of rank 1 hashes
- Top rank will end up in extreme case as single hash Ω but normally probably few hashes
- Packets carrying all this will be called Hierarchical SNPs



Harder Part: Partitioning

- CSNP do communicate absence, not only presence
  - Only detectable if 2 ranges of elements can be compared
  - Range implies ordering (ideally total ordering)
- Back to sqrt(1), we need CSNP like ranges for HSNP checksums
- Given limited packet size there will be multiple HSNP packets in flight, each with a range in it
- Gotcha: Hashes only easy to compare if ranges match

Bag of tricks

- HSNP ranges go from system-ID to system-ID (including pseudo nodes, thanks Les ;-)
- Infeasible to ”synchronize ranges with peers”
  - It would precondition a complex protocol to ensure e’one’s on the same page, basically partition sync
- Everything above rank 0 is using XOR of the set (only known practical self-inverse hash function)
  - Trivial to keep cheap “on the fly per system-ID” hash value during LSDB changes
  - Possible to “recalculate mismatched ranges” easily
    - E.g. received range is somewhat “longer”, just XOR in the system-ID hashes until range hit
- Loosely packing
  - Put several system IDs into the same “range” up to 50% of CNSPs generating a “packet”
  - Slide ranges are similar to fragments, i.e. on “packet size overflow of CSNPs” move system IDs to next higher range and so on recursively on “overflows”

Synchronization

- Nodes exchange in hellos maximum rank they are willing to support
- Perfect of all worlds normal exchange has 1 packet of high order with few hashes neighbors agree on
- Less perfect world will mismatch one of the hashes in N+1 rank
  - Node detecting responds with N rank packets covering the range in mismatched
  - Recursive “ping pong” until the slice of the database degrades into CSNPs and resulting flooding
    - This will need some kind of pacing mechanism
    - And some prevention from both sides going into disaggregation frenzy

Further Considerations

- Should we use standard CSNPs right after adjacency bring up
  - Less efficient, safer
- Something will need to be said about broadcast adjacencies
- Should we use 2 hash functions and XOR results to guard against hash attacks/degeneration
- Or checksums (much more computation but order influences computed value)