802.1Qbp – ECMP
Multicast Load Spreading

Ben Mack-Crane
(ben.mackcrane@huawei.com)
Observations on Multicast ECMP

• Multicast cannot use the unicast load spreading mechanism
  – Must forward on multiple ports (cannot select just one)
  – Random selection & replication can lead to duplication & loops

• ECMP for unicast traffic makes congruence (unicast-multicast and bi-directional) either easy or impractical (depending on how the definition is adjusted)
  – In either case congruence is not a concern with ECMP

• Multicast traffic must be constrained to a tree
  – to avoid loops and duplicate frames
Spreading Multicast Traffic

• In SPBM each service instance (I-SID) has distinct group addresses used to carry client multicast/broadcast traffic
  – Group addresses composed from SPSourceID & I-SID
  – # multicast flows = #service instances * #edge nodes
• Multicast filtering governed by VID and address (not Flow ID)
• Each multicast address can be independently routed
• Could assign each address to a different SPT
  – All nodes must agree on assignment to produce consistent forwarding state
  – Potentially large calculation (tree per address)
  – Probably more addresses than SPTs anyway
One Approach – Hashed SPT per Source

• Select “random” tree from SPT set for each source node
  – Select from all SPTs, not just those selected by .1aq tie-breakers
• Use this tree for all flows from that node
  – All I-SID multicast from source node use same tree
  – I-SIDs have varied endpoints, so some spreading within tree
• Use hash (e.g., FNV) to select one “parent” from set of equal cost parents calculated for unicast ECMP
  – Modest addition to route calculation
  – Include source node MAC address in hash to create variation

• Tried this out in an SPB simulator…
Unicast SPB, e.g. between 26 and 32

SPB selects a single path using an ECT tie-breaking function.
Unicast ECMP, e.g. between 26 and 32

ECMP load spreading utilizes all links on equal cost paths for unicast traffic.
SPB Multicast Tree, e.g. I-SID 255 from 26

Multicast selects links from one equal cost tree using ECT tie-breaker.
Multicast load spreading selects links from all equal cost paths using a hash function (in this case FNV).
Code for Parent FNV hash

```c
#define C1AQ_SYST_HASH_PARENT(result, syst, r, n)
{
    register tUINT32 hash = 0x811C9DC5;
    register tUINT64 fodder;
    register tUINT32 fnvPrime = 0x01000193;
    register tUINT32 best = 0;
    register int k, m, np = syst->node[n].np;
    for (m=0; m<np; m++)
    {
        fodder = syst->node[r].sysIdMac[0];
        for (k=0; k<7; k++)
        {
            hash = hash ^ (fodder & 0x000000ff);
            hash = hash * fnvPrime;
            fodder = fodder >> 8;
        }
        fodder = syst->node[syst->node[n].node[m]].sysIdMac[0];
        for (k=0; k<7; k++)
        {
            hash = hash ^ (fodder & 0x000000ff);
            hash = hash * fnvPrime;
            fodder = fodder >> 8;
        }
        if (hash > best)
        {
            best = hash;
            result = m;
        }
    }
    result = (m==0 ? -1 : syst->node[n].parent[result]);
}
```

This is the code in the SPB simulator used to generate these slides – I’m not sure this is a correct implementation of FNV – comments welcome!

- Random parent selection from ECMP set to produce source tree
- Uses Highest Random Weight (RFC 2991) to minimize impact of topology change
All SPB Multicast Trees, e.g. I-SID 255

Set of multicast trees are congruent.
All Multicast Trees, e.g. I-SID 255

Multicast load spreading selects links from all equal cost paths using a hash function (in this case FNV). Different trees are selected for each root by including root MAC address in hash.
Observations on this Approach

• ECMP algorithm used for both unicast and multicast
  – Provides load spreading for both types of traffic
• Multicast spreading uses a standard hash (pseudo-random)
• Good computational performance (relatively minor change)
• No provisioning required! (just like unicast)
  – No selection or configuration of VID or tie-breaker needed
• Propose further study of spreading performance and selection of a standard hash algorithm for use in multicast route calculation
One Concern – Multicast State Scaling

• Feedback expressing concern about scaling of multicast state
  – Multicast state is required per group address (I-SID endpoint)
• In networks with many BSIs with many endpoints each…
  – Result is many many group addresses registered in FDB
• E.g., virtual desktop VLAN may have 100s of endpoints (1000’s of users)
  – With default .1aq this means 100s of group addresses
  – And that is just for one I-SID!
• In large DC networks many group addresses may be assigned to the same tree (many more addresses than trees)
• Can we provide better scaling behavior?
Loop Free SPT Set

- Data center “fat tree” network architecture has a very regular structure
- A shortest path tree can match an SPT Set (i.e., be SPT from all endpoints)
- Using a shared tree for multicast reduces the forwarding state required (i.e., can use one address per service instead of one address per service endpoint)
ECMP with Shared Trees

- Shortest path trees rooted at spine nodes can form a balanced cover set
- Load spread by random assignment of each service instance to one of the shared trees
- Can realize significant reduction in multicast state (e.g., order of magnitude or more)
Observations on Shared Trees

• .1aq ECT Algorithm knobs may be used to tune trees
  – Create a set of trees that use all links
  – Each link used by the same number of trees (absent faults)
• VIP Default Backbone Destination address default is a single value per I-SID (BSIGA)
• Worthwhile to study shared trees and the options for supporting this feature
Multicast ECMP in 802.1Qbp

• So far in Qbp we have discussed the following:
  – Treat multicast the same as in .1aq (one congruent SPT set)
  – Provision multiple .1aq SPT Sets (tie-breakers) in one VLAN
  – Automatic selection from all possible SPTs, one per source node
  – Support shared trees to address FDB scaling issues

• These are four out of many possibilities
• Need to consider benefits of supporting various options
  – Better spreading characteristics
  – Less configuration (e.g. fully automatic)
  – Better fit with existing standards
  – Ability to control traffic placement when needed
ECMP Multicast Attributes

• **Granularity of SPT selection?**
  – One (per region)
  – One per source node
  – N per source node
  – One per address

• **How many SPTs in selection set?**
  – One .1aq tie-breaker subset
  – N .1aq tie-breaker subsets
  – All SPTs

• **How many group addresses?**
  – One per I-SID endpoint
  – One per I-SID (requires shared tree)

• **Selection of SPT**
  – Automatic (requires standard hash)
  – Provisioned (may require ISIS-SPB extension)

• **Assignment of I-SID to SPT**
  – Automatic (requires standard hash)
  – Provisioned (may require ISIS-SPB extension)
Some Possible (Desirable?) Combinations

• **All Automatic:** maximize number of trees, spreading opportunity
  – All SPTs (e.g., hash selection from ECMP)
  – One SPT per source node (to keep computation tractable)
  – One address per I-SID endpoint (so shared trees are not required)

• **All Provisioned:** minimize options, maximize control
  – One .1aq tie-breaker subset
  – One SPT per source node
  – One address per I-SID endpoint

• **Minimize multicast FDB state**
  – N .1aq tie-breaker trees (to provide cover set)
  – N SPTs per source node
  – One address per I-SID (requires shared tree)
  – Provisioned (or Automatic?)

Need to choose combinations to support in 802.1Qbp.