Policy-Aware Multi-Path Inter-Domain Path Computation in NS-3 Network Simulations

Jarno Rajahalme

Nokia Siemens Networks

ICT SHOK FI Results Seminar 15.2.2011

Nokia Siemens Networks

Outline

- Motivation
- NS-3 Code Improvements for Large Topologies
- New/Updated NS-3 Components
- Inter-Domain Routing Topology
- Valley-Free vs. BGP Policy
- BGP Routing Policies
- BGP Path Computation Emulation
- Algorithm
- Equal-Cost Multi-Path (ECMP)
- Next Steps

Motivation

- Network architecture development needs simulation tools
  - Inter-domain routing
    - Mapping systems
  - Data-oriented/content-centric rendezvous/resolution systems
  - Multi-Path TCP
- Current tools not scalable to large inter-domain topologies
  - Never tested with large topologies
  - Routing models not scalable
- No inter-domain routing models
  - Only shortest path, disregarding all inter-domain policies
- No multi-path support
  - However, a lot of current development on multi-path
- Need tools that can be shared with research partners
  - EU projects, universities

NS-3 Code Improvements for Large Topologies

1. Change from linear search to either binary search or associative maps
2. Fix nix-vector code to properly handle longer than 32-bit vectors
3. Defer clearing out caches after topology changes
4. Cache vector of neighbor nodes for later use
5. Cache a set of reachable intra-domain and sibling nodes

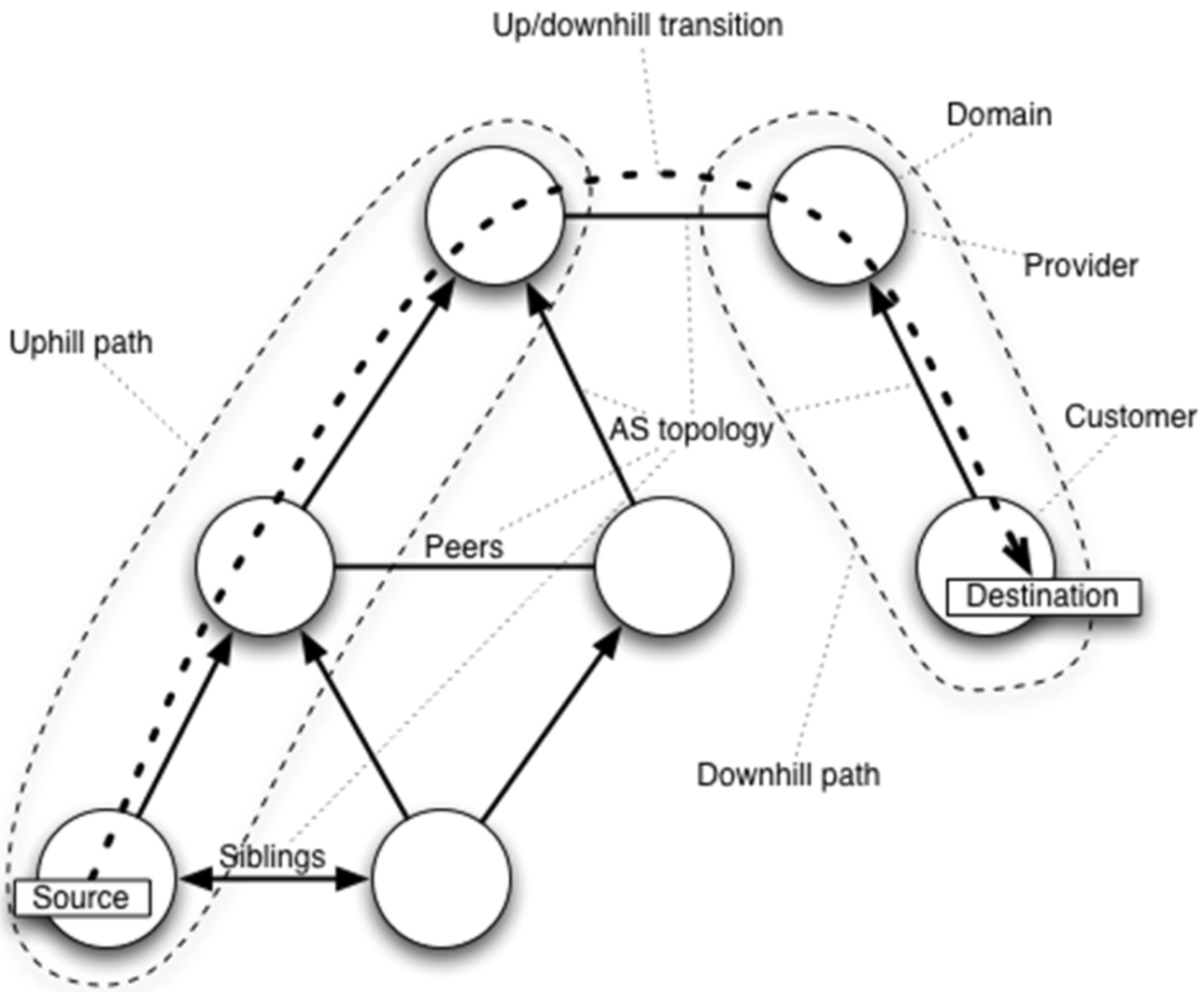
New/Updated NS-3 Components

1. CAIDA Topology reader
2. Coding of inter-domain relationships to IP interface metric

Relationship	Local metric	Remote metric
Link to a customer	0	>0
Link to a peer	0	0
Link to a provider	>0	0
Link to a sibling	>1	>1
Intra-domain link	1	1

3. AS Number Attribute for the `Node` class
4. Inter-domain path computation

Inter-Domain Routing Topology



Note: Uphill and downhill graphs depend on source and destination only, respectively!

Valley-Free vs. BGP Policy

- All policy-compliant paths are valley-free (>99%)
- All valley-free paths are NOT policy-compliant
  - Only a subset is
  - Esp. shortest valley-free paths are not guaranteed to be policy-compliant
    - BGP policy path inflation ~25% w.r.t. shortest valley-free
- Need to emulate BGP policies to find the shortest policy-compliant path
  - Policy to be evaluated at each AS, working towards the destination
    - Which of the available paths is advertised upstream (towards the source)?
    - Which of the advertised paths is selected?

BGP Routing Policies

In the order of decreasing preference:

1. Customer paths (*for additional revenue*),
2. Paths to siblings' customers (*revenue for siblings*),
3. Direct peering paths (*revenue neutral*),
4. Paths via sibling's peers (*cheaper than transit*),
5. Direct provider paths, and finally, if nothing else is available,
6. Provider paths via siblings (*more internal costs*).

- AS-path length as a tie-breaker (only!) *within each case*
- Path "cost" internal to each AS
  - Neighbors see only the advertised paths (if any) and the path AS-length
- Ordering can be tuned with actual link cost attributes
  - E.g., some sibling links can be expensive, or some provider links can be cheaper than others

BGP Path Computation Emulation

- Use path costs to choose exported paths
- Two loop prevention modes:
  - Intra-domain: Choose shortest path within the domain (incl. sibling ASes)
  - Inter-domain: Explore all possible paths, but do not loop to siblings
- Aggressive pruning
  - Path exploration is terminated on branches with path cost exceeding known upstream minimum
    - E.g., if peering path exists, do not explore providers
- Algorithm outline
  - Compute Destination DownGraph
  - Explore Source UpGraph until find nodes on the destination downgraph
    - Calculate path cost at each AS
    - Export least-cost paths to upstream ASes

Algorithm

Algorithm 4: Find the shortest policy-compliant path length.

Data: AS graph G, customer links C, sibling links S, peering links P, link costs LC, source S_{AS}, destination D_{AS}

Result: The shortest policy-compliant path from S_{AS} to D_{AS}

```
begin
    Up ⟵ UpGNode(S_AS)                          // Uphill nodes
    Dn ⟵ GetDownGraph(G, D_AS)                  // Downhill nodes
    if S_AS ∉ Dn then // not direct customer
        while node ⟵ Up.GetNextToExplore() do
            for ∀j : (node, j) ∈ G − C do // uphill neighbors
                if Up.IsLoop(j) then
                    continue                    // next neighbor
                if j ∈ Dn then // path found?
                    if (node, j) ∉ S then // not sibling
                        Up.InsertBack(j, LC_(node,j), Dn.cost(j))
                    else
                        Up.UpdateCost(j, LC_(node,j), Dn.cost(j))
                        Up.InsertFront(j, LC_(node,j), Dn.cost(j))
                else if (node, j) ∉ P then // explore later
                    if Up.CheckCost(j, LC_(node,j)) then
                        if (node, j) ∈ S then // sibling
                            Up.InsertFront(j, LC_(node,j))
                        else
                            Up.InsertBack(j, LC_(node,j))
    return Up.Path(Dn)
```

Equal-Cost Multi-Path (ECMP)

- Ref. RFC 2992
- Keep track of multiple equal-cost next hops
- Choose a next hop for each packet
  - Want no TCP reordering
    - Choose via a hash on the 5-tuple
- Implemented in NS-3
  - “Hash” on source node ID and destination IP address
    - Same source/destination pair will always use the same path

Next Steps

- Polishing for NS-3 submission, three changesets
  1. Bug fixes
  2. Performance improvements
  3. Routing code
  4. API for ECMP, e.g. via attributes?
  - No need to maintain compatibility with quarterly NS-3 releases
  - Can be used as a platform for inter-domain distributed system simulations
    - EU projects
    - TEKES projects
    - University collaboration
    - Internal projects
- nix-vector -> Pathlets?