Towards More Adaptive Internet Routing

Mukund Seshadri
(mukunds@cs.berkeley.edu)

Prof. Randy Katz

Motivation – Inter-domain Routing

- Inter-domain routing failures often last several minutes [Labovitz et al.]
  - Slow BGP convergence
  - Can take up to 15 min to recover
- Reachability failures can often be circumvented by using alternate routes
  - E.g. 12-Node RON recovered from 32 outages over 64hrs, Mar'01
  - [Feamster03] estimates recovery from 50% of failures.
  - Overlays were used – small-scale solution only.
- Can we modify inter-domain routing to make alternate routes available (when present)?

Motivation – Intra-domain Routing

- Typically link weights (OSPF) set to achieve desired utilizations (for known traffic matrix)
- Can Performance be a problem?
  - [Sprint02] reported that at any given time, some link in the network was likely to be "over-loaded" (>50% utilization).
- Cannot adapt to changes in traffic load
  - Currently addressed by heavy over-provisioning

- If such over-povisioning is not affordable: can we automatically adapt to significant changes in the load?

Part 1- Inter-domain Routing

- Goal – improve inter-domain reachability by using existing redundancy in the AS graph.
- Network-layer approach -
  - Provide alternate routes via an extension to the (BGP) path vector protocol
- Evolutionary/Overlay approach (last retreat)
  - Improve scalability using using topology information (about route diversity).

Path Vector Background

- Each node A advertises to each neighbour B: the cost of A's shortest route to each destination (prefix), and the list of nodes (ASes) on that route.
  - B selects the shortest of all the routes to a particular destination that received from its neighbours, after adding the cost of the link between B and its neighbour

- Prior work has designed protocols for maximally disjoint multi-path distance vector protocols [JJ89]
  - However, all possible paths are explored => unnecessarily high overhead
  - Does not use path vector information => complex and slow.

Our Network Layer Approach

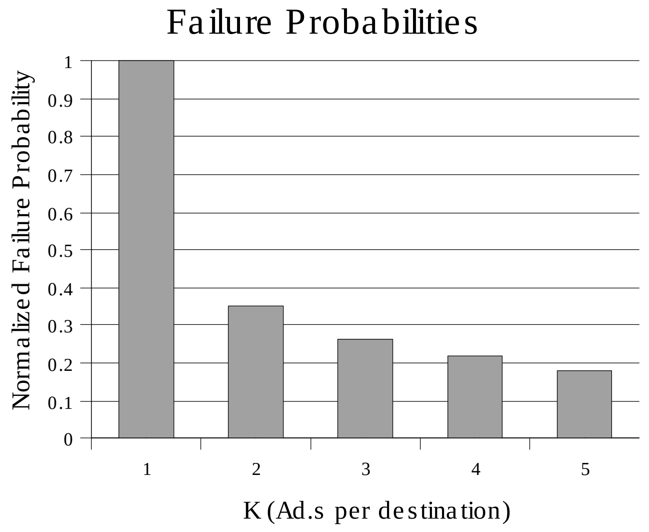
- Extend BGP's path vector protocol to advertise *k* (~2) routes per destination instead of 1.
  - Factor k increase in advertisement overhead
- First of the k routes is computed using the current BGP route-selection.
- The remaining k-1 routes are selected to be maximally link-disjoint (at the AS-level).
  - Sequential greedy selection of routes
  - Heuristic to reduce probability that a change to the default route will be accompanied by a change to the alternate routes
  - Assumes random single-link failure.

Service Model

- How will the use of the alternate routes be triggered?
  - Network node can automatically switch when the default route changes;
    - ..until the default routing entry stabilizes
    - ..not a complete solution
    - ..combine with BGP-RCN?
  - Best way to validate a routing entry is to send and receive packets via that route
    - End-hosts already do this – can indicate reachability failures via a flag in the packet-header.
    - Flag essential for loop-avoidance.

Results

- Construct AS-level topologies and default paths from BGP Routeviews data.
  - Inaccuracies due to symmetry assumption and hidden edges
  - ~500 nodes, 100 src/dest.
- Construct routing tables using *k*-path vector.
- Find reachability/failure probability of all destinations for a given node (under random single link failure)



- Just using *k*=2 greatly improves reachability

Part 2 – Intra-domain Routing

- Goal: load-sensitive dynamic routing –
  - Assumptions:
    - traffic changes faster than traffic engr.'s timescales (~10'sof minutes-hours).
    - Heavy overprovisioning (2x) is not feasible.
  - Packet-switching, no reservation-based models
    - Does not change the interface to end-hosts or other networks
- Issues:
  - Stability can be hampered by herd behaviour and use of stale information
  - Load-balancing
    - One route per destination (currently used) makes this harder

Background

- Typical intra-domain protocol: link-state, e.g. OSPF
  - Current state/cost of each adjacent link is reported by each node to all other nodes
  - A shortest-path algo. (Dijkstra) is then used by each node to compute one route per destination.
- Using a load-based metric (like delay) directly can be unstable [Khanna89, Srihari99]
  - [Khanna89] proposes a metric that resembles delay-based at low utilization and capacity based at high utilization.
    - Still can see oscillations, doesn't balance load (related work: [Wang92]).
    - Network/Diameter-dependent metric-setting

Points of Attack

- Granularity of Routing Unit
  - Currently one route per destination
- Route Selection Method
  - Currently least-cost first (“greedy”)
- Routing Metric
  - Currently static metrics reflecting hop-count or weights pre-determined by traffic engineers.

Our approach – Route granularity

- One route per destination network node => high-volume unit of re-routing => harder to load-balance

- Therefore a node A divides the traffic through it to a particular destination node into *B* buckets
  - Division into buckets done independently by the network node (hash of src/dst address), thus not affecting the interface to other networks or end-hosts.
  - Small B desirable to avoid per-flow routing state.

- One route is maintained for each bucket

Our Approach – Randomization

- Assume the link state (metric) is the load in the last link state period.
- Link state is inherently stale
  - This can cause herd behaviour, leading to instability and imbalance

- We introduce randomness into the routes selected across different buckets for the same destination
  - Randomly choose from r best routes.
  - Best of r random routes (selected proportional to static costs)
  - [Mitzenmacher97] showed that "best-of-2 random selection" was ideally suited for server load-balancing with stale info.
- Can also randomize time of route-change across buckets
- Edge-based route-selection, to avoid routing inconsistency.

Our Approach – Metric

- Separation of static and dynamic metrics
  - Capacity (or propagation delay) can be advertized infrequently
  - Load (or queue-length) need more frequent advertisement
- Load metric can be further improved (future work)
  - By simulating the system and building a model of load-transition.
  - This improves performance for best-first selection, but not significantly for random selection with “bucketization”

Simulation Results

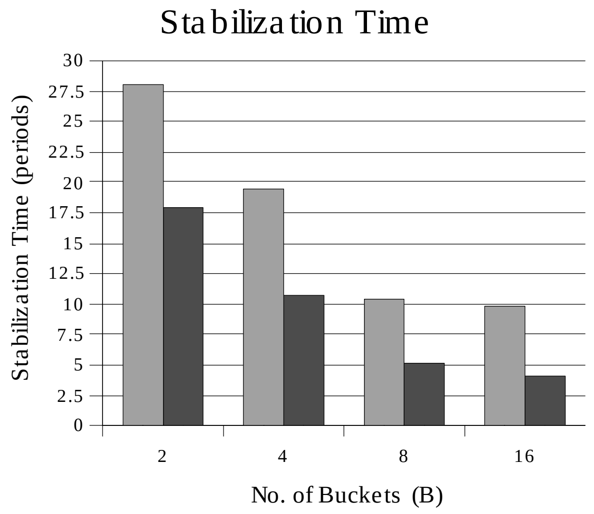
- Use random “fork”/t-s topologies (~50 nodes)
  - Flow-level, assume the capacities and incoming loads to links can be reported.
- Traffic matrix such that the overprovisioning factor O.F. (min[Capacity/Load]) is low (~1.2)

- Objectives
  - low stabilization times from initial state, or after increases in link loads (factor of 2 or more)
  - Low loss-rates assuming continuous arrival/departure of end-host flows

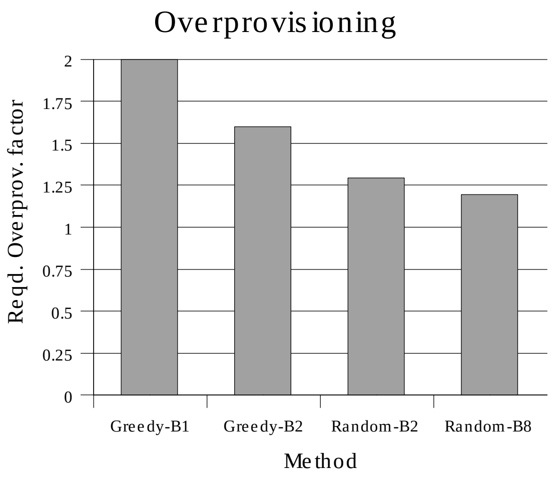
Results

- "Bucketization" improves stabilization times (and loss rates) even with moderately low values of ***B***
  - Assuming randomization of time of route-change
  - Since the unit of traffic change becomes significantly lower than total link loads.

- Random selection is a significant improvement over best-first selection.



Random: Red bars
Best-first: Blue bars



Conclusion and Future Work

- In Conclusion..
  - Can improve resilience of inter-domain routing by making alternate routes available at the network layer
  - Can make intra-domain routing more adaptive to load (and therefore require lower over-provisioning), by using per-bucket routes and random route selection.

- Future Work:
  - Better, Dynamic Evaluation Scenario
    - Failure location/time data for inter-domain routing
    - Traffic matrix and topology for intra-domain routing
  - Better metric for load-sensitive routing
    - Use model of state change.
    - Effect of filtering, incorporate delay info.