Advanced Topics in Routing

EE122 Fall 2012

Scott Shenker

http://inst.eecs.berkeley.edu/~ee122/

Materials with thanks to Jennifer Rexford, Ion Stoica, Vern Paxson and other colleagues at Princeton and UC Berkeley
Holy Trinity of Routing: LS, DV, PV

• Normally presented as the complete story

• But we know how to do much better

• That is what we will talk about today…. 
Major Routing Challenges:

- Policy Oscillations
- Resilience
- Traffic Engineering
Another Purpose for Today

• EE122 (CS version) is algorithmically vacuous
  – AIMD is the high point of intellectual depth (ugh)

• The algorithms described today are nontrivial
  – Algorithms simple, but their properties are nonobvious

• You will prove two results as a class exercise
  – 5 minutes, in groups, try to come up with reasoning
  – I’ll help shape it into a proof
Policy Dispute Resolution
Policy Oscillations

• Last time we discussed how BGP might never converge due to “policy oscillations”

• We now discuss how we might solve this problem
Policy Oscillations (cont’d)

• Policy autonomy vs network stability
  – Oscillations possible with small degree of autonomy
  – Focus of much recent research

• Not an easy problem
  – PSPACE-complete to decide whether given policies will eventually converge!

• However, if policies follow normal business practices, stability is guaranteed
  – “Gao-Rexford conditions”
  – Essentially the provider/peer/customer policy categories
Theoretical Results (in more detail)

• If preferences obey Gao-Rexford, BGP is safe
  – Safe = guaranteed to converge

• If there is no “dispute wheel”, BGP is safe
  – But converse is not true

• If there are two “stable states”, BGP is unsafe
  – But converse is not true

• If domains can’t lie about routes, and there is no dispute wheel, BGP is incentive compatible
Objectives for New Policy Approach

• Do not reveal any ISP policies

• Distributed, online dispute detection and resolution

• Pick “normal” path (according to policies) if no oscillation exists
  – *Get something reasonable if oscillation would exist*

• Account for transient oscillations, don’t permanently blacklist routes
Example of Policy Oscillation

“1” prefers “1 3 0” over “1 0” to reach “0”
Step-by-Step of Policy Oscillation

Initially: nodes 1, 2, 3 know only shortest path to 0
1 advertises its path 1 0 to 2
Step-by-Step of Policy Oscillation
3 advertises its path 3 0 to 1
Step-by-Step of Policy Oscillation
1 withdraws its path 1 0 from 2
Step-by-Step of Policy Oscillation
Step-by-Step of Policy Oscillation

2 advertises its path 2 0 to 3

advertise: 2 0
Step-by-Step of Policy Oscillation
3 withdraws its path 3 0 from 1
Step-by-Step of Policy Oscillation
Step-by-Step of Policy Oscillation

1 advertises its path 1 0 to 2
Step-by-Step of Policy Oscillation
Step-by-Step of Policy Oscillation

2 withdraws its path 2 0 from 3

withdraw: 2 0
Step-by-Step of Policy Oscillation

We are back to where we started!
Nodes See Signs of Trouble

• Route choices oscillation
  – Node 1:
    o 1 0 , 1 3 0 , 1 0 , 1 3 0 , .....  
  – Node 2:
    o 2 0 , 2 1 0 , 2 0 , 2 1 0 , .....  
  – Node 3:
    o 3 0 , 3 2 0 , 3 0 , 3 2 0 , .....  

• Choices alternate between more preferred and less preferred routes
Basic Idea

• If node notices that it is constantly selecting routes that are more / less preferred than previous route
  – Node thinks it may be involved in oscillation

• Computes local “precedence” figure
  – Higher precedence value for less preferred routes
  – In example, 1 0 gets higher value than 1 3 0

• Route advertisements carry this precedence
  – Two precedence values:
    o Incoming (carried by packet)
    o Local (determined by own past history)
Precedence Calculation

• Routes are first ranked by “incoming precedence”
  – Pick most preferred route among those with lowest incoming precedence value

• Outgoing precedence is sum of incoming and local precedence
Use of Precedence Values

• Maintain *history* of routes encountered during oscillations
  • In table below, prefer P0 to P1 to P2

<table>
<thead>
<tr>
<th>AS Path</th>
<th>Incoming Precedence</th>
<th>Local Precedence (Computed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>P1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

• Pick path P2, mark with precedence 1
Example of Policy Oscillation
Step-by-Step of Policy Oscillation

1 advertises its path 1 0 to 2
Step-by-Step of Policy Oscillation
3 advertises its path 3 0 to 1

Step-by-Step of Policy Oscillation
Step-by-Step of Policy Oscillation
1 withdraws its path 1 0 from 2
Step-by-Step of Policy Oscillation
Step-by-Step of Policy Oscillation

2 advertises its path 2 0 to 3

Routes stabilize at this point
3 cannot choose 2 0 route from 2, because of higher precedence value

advertise: 2 0

Precedence 1
“Proof” of why it works

• Assume that policy oscillation exists within scheme

• Some router A must prefer a path offered by a router B that does not prefer that path
  – If everyone is getting first choice, no oscillation!
  – So A’s first choice is B’s second choice for some A,B

• But router A cannot choose that path because it will have a lower precedence
Properties of Solution

• If no policy oscillation exists, get usual routes
• If policy oscillation would have existed, approach short-circuits oscillation
• If, after convergence, non-zero global precedence values exist, $\Rightarrow$ dispute(s) exist
• Only precedence values advertised, no other routes or policies revealed
• Why isn’t this deployed?
Routing Resilience
Resilience

• Basic routing algorithms rely on timely consistency or global convergence to achieve ensure delivery
  – LS: routers need to have same picture of network
  – DV: if algorithm hasn’t converged, might loop

• As nets grow, this gets harder and takes longer
  – *Need both consistency/convergence and timeliness!*

• Creates lag between failure detection and recovery
  – Lag is biggest barrier to achieving 99.999% reliability
Hacks Used Today

• Preconfigured backup paths
  – When link fails, router has a backup route to use
  – Very helpful against single failures
  – Only limited protection against multiple failures
  – No systematic paradigm

• ECMP: Equal-Cost Multipath
  – Similar to backups, but narrower applicability
  – Choose among several “shortest-paths”
Solutions Presented Today

• Multipath (one slide)
• Failure-carrying packets
• Routing-along-DAGs
Multipath Routing

• Multipath:
  – Providing more than one path for each S-D pair
  – Allow endpoints to choose among them
    o This can be implemented by having a “path” field in packet

• Good: if one path goes down, can use another

• Bad: Delay while endpoints detect failure (RTT)

• Absolutely necessary because of E2E arguments
  – But not a fundamental paradigm shift

• Part of solution, but still need more reliable routing
Can we completely eliminate the need to “reconverge” after link failures?

i.e., can we tolerate failures without losses?
Failure-Carrying Packets (FCP)
**FCP Approach: Step 1**

- Ensure all routers have consistent view of network
  - But this view can be out-of-date
  - *Consistency is easy if timeliness not required*

- Use reliable flooding
  - Each map has sequence number

- Routers write this number in packet headers, so packets are routing according to the same “map”
  - Routers can decrement this counter, not increment it
  - Eventually all routers use the same graph to route packet

- This achieves consistency, but not timeliness….
FCP Approach: Step 2

• Carry failure information in the packets!
  – Use this information to “fix” the local maps

• When a packet arrives and the next-hop link for the path computed with the consistent state is down, *insert failure information into packet header*
  – Then compute new paths assuming that link is down

• If failure persists, it will be included in next consistent picture of network
  – Then not needed in packet header
Example: FCP routing
Example: FCP routing
Class Exercise: Prove This Works

• Develop line of argument about why this guarantees connectivity

• Under what circumstances does guarantee hold?
Keys to Proof

• Deadend: as long as map plus failures has connectivity, no dead ends

• Loops: Assume loop. The nodes on the loop all share the same “consistent” map plus a set of failures in the packet header. Therefore, they compute the same path. Contradiction.
Condition for Correctness

• Consider a set of changes to network from the last consistent map before packet is sent until TTL of packet would expire.

• If intersection of all network states during change process is connected, then FCP will deliver packet
Properties of FCP

• Guarantees packet delivery
  – As long as a path exists during failure process

• Major conceptual change
  – Don’t rely solely on protocols to keep state consistent
  – Information carried in packets ensures eventual consistency of route computation
  – This theme will recur in next design….
  – Ion’s Stoica’s thesis!
Results: OSPF vs. FCP

- Unlike FCP, OSPF cannot simultaneously provide low churn and high availability.
Results: Backup-paths vs. FCP

- Unlike FCP, Backup-paths cannot simultaneously provide low state and loss rate.
Problems with FCP

• Requires changes to packet header
  – And packet headers could get long

• Requires fast recomputation of routes
  – Can precompute common cases, but worst case is bad

• Does not address traffic engineering
  – What is that?
Traffic Engineering (TE)

• Connectivity is necessary but not sufficient

• Need to also provide decent service

• Requires that links on the path not be overloaded
  – Congestion control lowers drop rate, but need to provide reasonable bandwidth to connections by spreading load

• TE is a way of distributing load on the network
  – i.e., not all packets travel the “shortest path”
Routing Along DAGs (RAD)
Avoiding Recomputation: Take II

• Recover from failures without global recomputation

• Support locally adaptive traffic engineering

• Without any change in packet headers, etc.

• Or requiring major on-the-fly route recomputation
Background

• Focus only on routing table for single destination
  – Could be a prefix, or a single address
  – Routing to each destination is independent, so this is fine

• Today we compute *paths* to particular destination
  – From each source to this destination there is a path

• When path breaks, need to recompute path
  – The source of all our troubles!
Our Approach: Shift the Paradigm

Routing compute paths from source to destination

Move from path to DAG (Directed Acyclic Graph)
If a link fails, all affected paths must be recomputed

P DAG

Packets can be sent on any of the DAG’s outgoing links
No need for global recomputation after each failure
DAG Properties

- Guaranteed loop-free
- Local decision for failure recovery
- Adaptive load balancing
Load Balancing

• Use local decisions:
  – Choose which outgoing links to use
  – Decide how to spread the load across these links
  – Push back when all outgoing links are congested
    o Send congestion signal on incoming links to upstream nodes

• Theorem:
  – When all traffic goes to a single destination, local load balancing leads to optimal throughput

• Simulations:
  – In general settings, local load balancing close to optimal
DAG-based Routing

• Essentially a principled paradigm for backup paths
  – Can tolerate many failures
  – Scalable
  – Easy to understand and manage
Computing DAG

• Use each link in a single direction
• DAG iff link directions follow global order
• Computing a DAG for destination \( v \) is simple:
  – Essentially a shortest-path computation
  – With consistent method of breaking ties
What about Connectivity?

- Multiple outgoing links improve connectivity
  - But can RAD give “perfect” connectivity?

- If all outbound links fail that node is disconnected
  - Even if underlying graph is still connected

- How can we fix this?
Link Reversal

• If all outgoing links fail, reverse incoming links to outgoing
RAD Algorithm

• When packet arrives, send out *any* outgoing link

• When an outgoing link fails (or is reversed)
  – If other outgoing links exist, do nothing
  – If no other outgoing links exist, reverse all incoming links
    o i.e., change them to outgoing
Link Reversal Properties

• Connectivity guaranteed!
  – If graph is connected, link reversal process will restore
    connectivity in DAG

• This has been known in wireless literature
  – Now being applied to wired networks

• If you don’t think this is neat, then you are asleep.
  – Local rule to produce ideal connectivity!
Class Exercise: Prove This Works

• Develop line of argument about why this guarantees connectivity
Keys to Proof

• Deadend: algorithm never results in dead-ends
  – At least one link will be outbound, if you have a link

• Loops:
  – Assume network does not have loop at beginning
    o (i.e., we have a DAG)
  – Link reversal cannot create a loop
    o Because reversed node cannot be part of a loop
  – Therefore, topology never in a state where a loop exists

• Are we done with proof?
No, link reversals might not terminate

• Must prove topology reaches fixed point
  – If underlying graph is connected

• Not reaching a fixed point means process of node reversals continues forever

• Since network is of finite size, this process must repeat in a cycle of node reversals

• How can we prove this is impossible?
Fact #1

• If a node has a path to the destination, then it will never reverse itself.

• Conclusion: the set of nodes with a path to the destination is nondecreasing
Fact #2

• For a node to do a second link reversal, all of its neighbors must have also reversed its links.

• Therefore, the set of nodes doing a link reversal is an expanding set.

• Can only re-reverse all reversing nodes if the process reaches the “edge” of network.

• But once this process touches a node which is connected to the source, it stops. QED.
Summary of RAD

• Local responses lead to:
  – Guaranteed connectivity
  – Close-to-optimal load balancing

• Can be used for L2 and/or L3
  – No change in packet headers
Why Isn’t RAD Enough?

• The link reversals are on the “control plane”

• They take time to compute

• Packets can be lost in the meantime…

• Exactly the problem with FCP route recomputation
  – Works on control-plane speeds, not data speeds

• Any suggestions?
Data-Driven Connectivity (DDC)

- Define link reversal properties in terms of actions that can occur at data speeds

  - Events: packet arriving in “reverse” direction
  - Action: remove that link from outgoing set

- Goal: define simple algorithms that can be supported in HW
  - Ask Panda for more details
Review

• Major Routing Challenges:
  – Resilience
  – Traffic Engineering
  – Policy Oscillations

• We have solutions for all of them!
  – FCP, RAD, and Policy Dispute Resolution

• Are they deployed? No…..
  – Will they be deployed? Maybe…..

• ..