Suffering Withdrawal

An automated approach to connectivity evaluation

Ben Dale, Nick Slabakov, Micah Croff, Tim Hoffman, Bruce McDougall

How do we deal with network failures today?

How do we deal with network failures today?

- Transport link fails?
- BGP session goes down facing transit?
- BGP session goes down facing CDN cache?
- Route withdrawn by routing protocol?

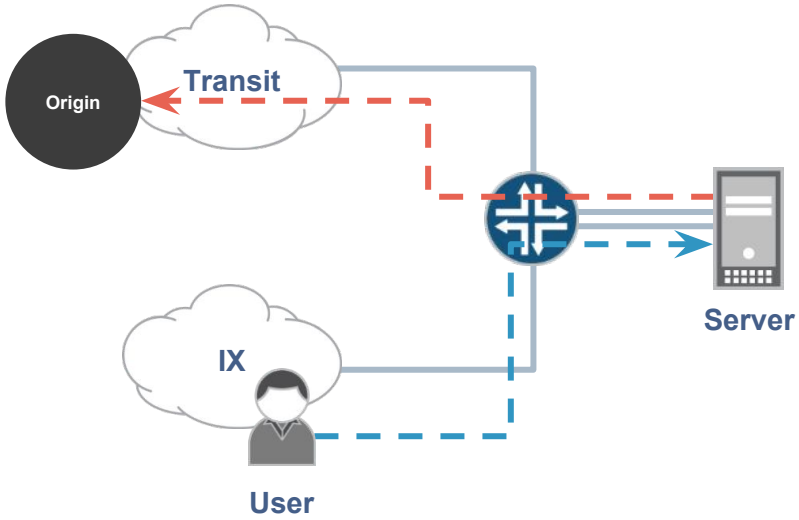
Routing will react to those changes, but the result is frequently insufficient or even undesired!

Where doesn’t this help?

The authors of this presentation have seen all these scenarios in their environments in the last couple of years

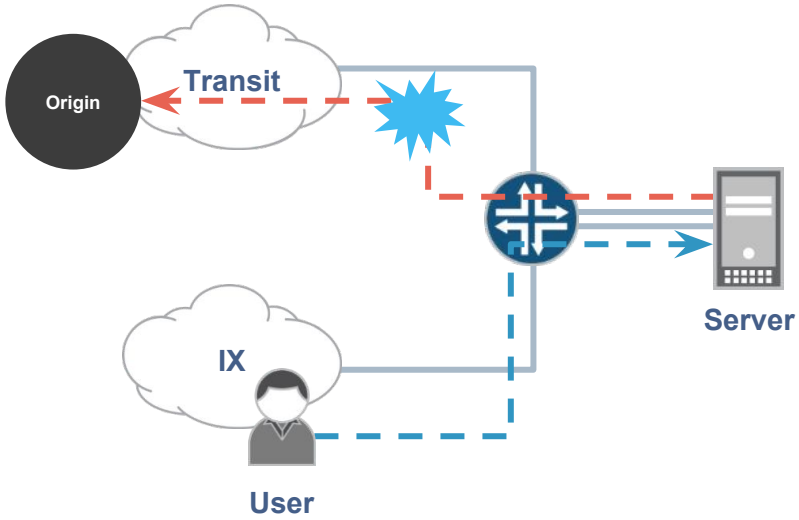
Transit provider fails for service node!

CDN node connected to peering and transit (CDN server in POP advertises anycast routes to router). Single transit provider



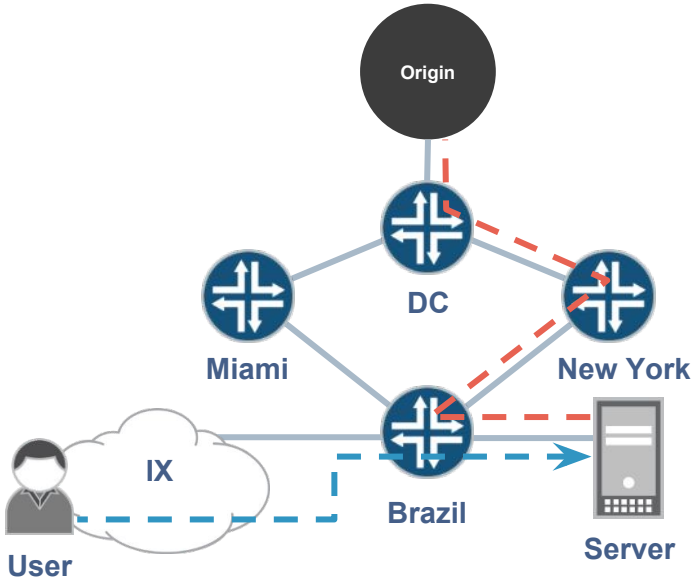
Transit provider fails for service node!

Transit provider fails. Origin calls from CDN server now failing. Anycast prefix from server still announced. User requests fail.



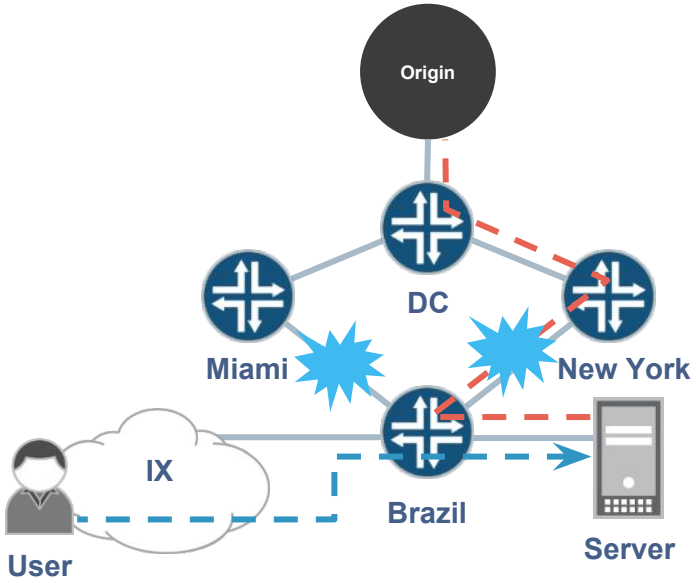
Service node is isolated from the backbone!

CDN node announcing anycast routes, and accessing origin via backbone.



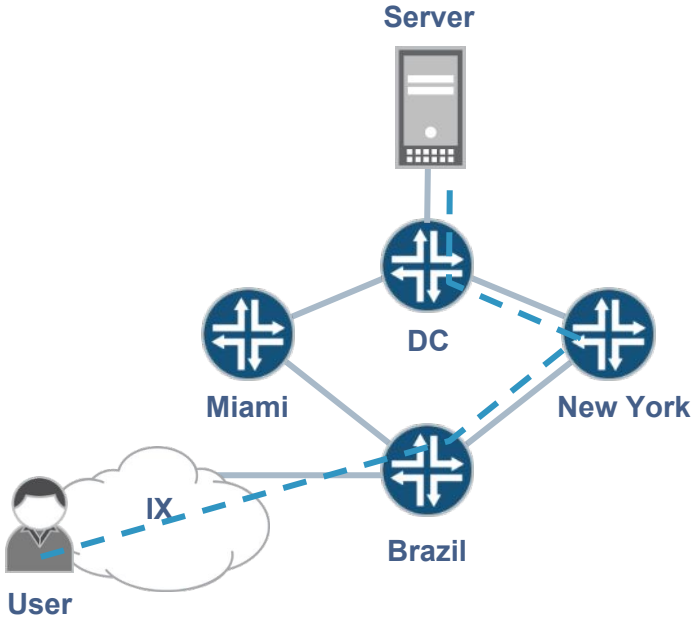
Service node is isolated from the backbone!

Backbone fails. Can no longer reach origin, but server is still announcing anycast CDN subnets. User requests fail.



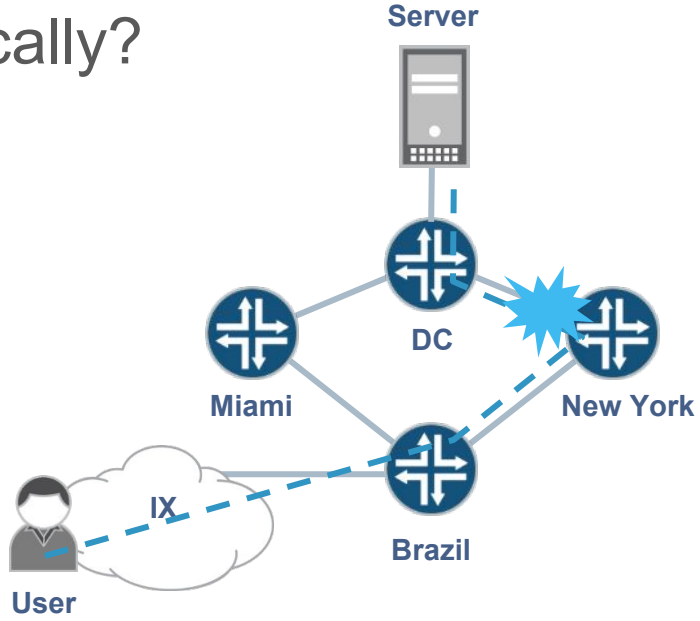
Backbone port starts blackholing traffic

User on IX accessing a service found over a backbone.



Backbone port starts blackholing traffic

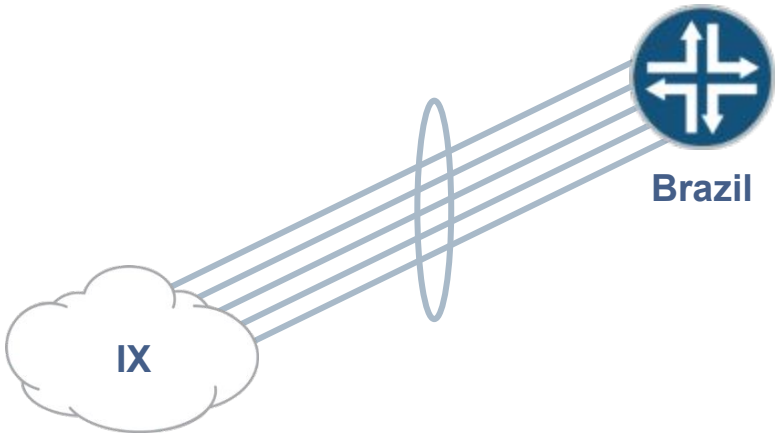
Line card starts blackholing traffic, but BGP/ISIS stays up. How do we detect this and withdraw routes advertised by Brazil automatically?



Need to shed traffic on LAG due to failed members

50Gbit LAG facing an IX. 40Gbit of traffic.

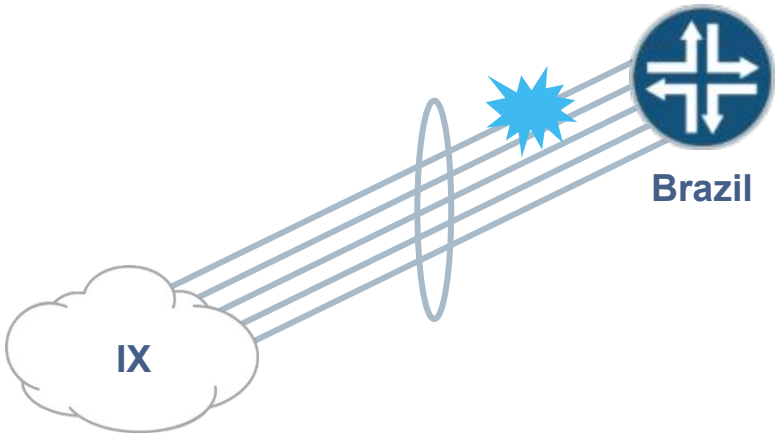
This is often solved with RSVP AutoBW (using aggressive preemption), however there is no equivalent functionality for EBGP



Need to shed traffic on LAG due to failed members

2 links fail - need to shed 10Gbit of traffic

How do we adjust BGP parameters to shed traffic off?



How do effective network operators solve this today?

Automate! Automate! Did I say Automate!

- Large operators solve this by taking routers and POPs out of service when they are in a degraded state.
  - Measure the health of a node using some arbitrary metrics and checks
  - When health = bad
    - Use a pre-defined process to automatically pass commands to “fix”…
      or
    - Take out of service and notify human

But where doesn’t this approach work?

- My router is still in service and peering with local networks, but;
  - Disconnected from backbone
  - OOB circuit is not reachable
- I don’t have OOB! (don’t build these networks!!!!)
- I haven’t automated remediation
  - Let’s now page a human. It’s 2am. Human doesn’t think straight, and takes 15mins to wake up, then 10mins to mitigate.

How else could we solve this?

Write custom scripts on routers

- Run scripts locally on routers
  - Vendor specific fixes - TCL/SLAX/etc?
  - Bash script running under crontab on the base OS of the router (BSD/linux)?

“Let’s count the number of prefixes in the default routing table and take this router out of service if it sees less than 400k routes”
“Let’s ICMP these 5 hosts. If 2 or more don’t respond for 5mins, take this router out of service”

We haven’t solved the issue of routers being aware of their actual ability to connect. Another router tells them what they are connected to, and they trust this following some policy processing.

Surely we can do better? (remember, we are in 2016 now!)

Can we learn something from VRRP?

- Track variables and weight them with a priority value
  - Prefix X in routing-instance Y
  - Physical state of an interface
- Assign different weights to different variables
- Reflect this into the VRRP priority
  - If our priority is now worse than other candidates for master, we will relinquish control the VIP

We propose a new method of conditional routing

No New Protocols

- No new protocols required
  - Leverage existing standards based protocols

- Leverage Existing TE Mechanisms
  - Local-pref, as-path prepending, apply policies (BGP)
  - Overload, Metrics/Costs (IS-IS)
  - Metrics/Costs (OSPF)

Weighted conditional routing to evaluate connectedness

- Find out how well connected I am
  - Is there packet loss to X
  - How many routes can I see based on Y policy
  - Other tests
- Take some actions on this basis
  - Take router out of service (via policy or overload)
  - Shed traffic
  - Stop transitive LSPs on a backbone traversing me

Buckets

Buckets

- Any number of named buckets (variables)
- Default value assumed to be 0, unless configured otherwise
- In general, the higher the value of a bucket, the more tests are passing, and the more connected/healthy a node is
- Buckets reference tests to define their values
- Buckets are used by routing configuration as a “from/match/if” attribute

Tests

Example Tests

- How many routes do we have matching X policy?
- How many routes in table X?
- How many routes from neighbour Y?
- What % of BGP peers in group X are established
- How many members are in this LAG?

Expected level is X;

+1 point to bucket A per Y routes/neighbors

+1 point to bucket A per Z% of neighbors established

Example Tests contd.

- How many members are up in this LAG?

For every member up in LACP, +10 points to bucket B (could be hard number or %)

- Can I ICMP to this IP?

For test X (based on RPM/IPSLA configuration) +10 points to bucket C when within SLA

Weighting Test Results

- For added flexibility, it should be possible to weight test results eg:

```
if (test A)
     then RED += 100
if (test B)
     then GREEN += 200
…
if (RED <=100)
     then action1()
else if (GREEN <=200)
     then action2()
```

Actions

Actions

- Routing policy from/match statement;

When bucket X is > or = or < a specified value, assume a positive match value

```
Policy term;
          from {
              condition-bucket GREEN below 5;
          }
          then {
              as-path-prepend "123 123 123 123";
          }
```

Actions

- Overload IGP

When bucket X is > or = or < a specified value, overload the protocol;

```
ISIS configuration;
        conditional-overload {
           condition-bucket RED below 5;
        }
```

Open questions

What additional tests or actions could we integrate?

- Fundamental aim is to build a framework that additional tests can be added to - we've started with
  - # of established BGP peers/full IGP neighbours
  - Presence of marker routes in BGP/IGP
  - # Active prefixes installed in named table
  - # Active next-hops for a given prefix
  - # Prefixes received from a given neighbor
- We would like to integrate anything that reasonably fits within this framework in our proposal.

How do we prevent cascading failures?

- If I don’t see this marker route in my IGP, panic and set the IGP to overload!
- If I don’t see ~600K routes in FIB, panic and apply REJECT-ALL to my export policy!
- What things can we always do without making things worse or creating cascading failures?
- Due to flexibility, most will be on users, but the hope is to use as-path prepends more than “rejects”

Should we use hard numbers or percentages?

- For most tests we could use a hard number of peers, routes, or LAG members, or a percentage
  - % of routes in table based on 24hr moving average
  - % of members in LAG
  - % of peers based on BGP group
- What is preferred?

Roadmap

Roadmap

Active involvement from 3 operators + Juniper and Cisco

- ~~Working group developing concept and testing~~
- ~~Present & solicit feedback at NANOG~~
- Submit as internet-draft to IETF and solicit feedback
- Progress towards RFC
- Vendor implementation

Thank you!
We would like your feedback

Tim Hoffman - tim@hoffman.net.nz
Ben Dale - ben.dale@gmail.com
Micah Croff - micahcroff@gmail.com
Nick Slabakov - slabakov@juniper.net
Bruce McDougall brmcdoug@cisco.com