

Google™ OpenFlow @ Google

Summary



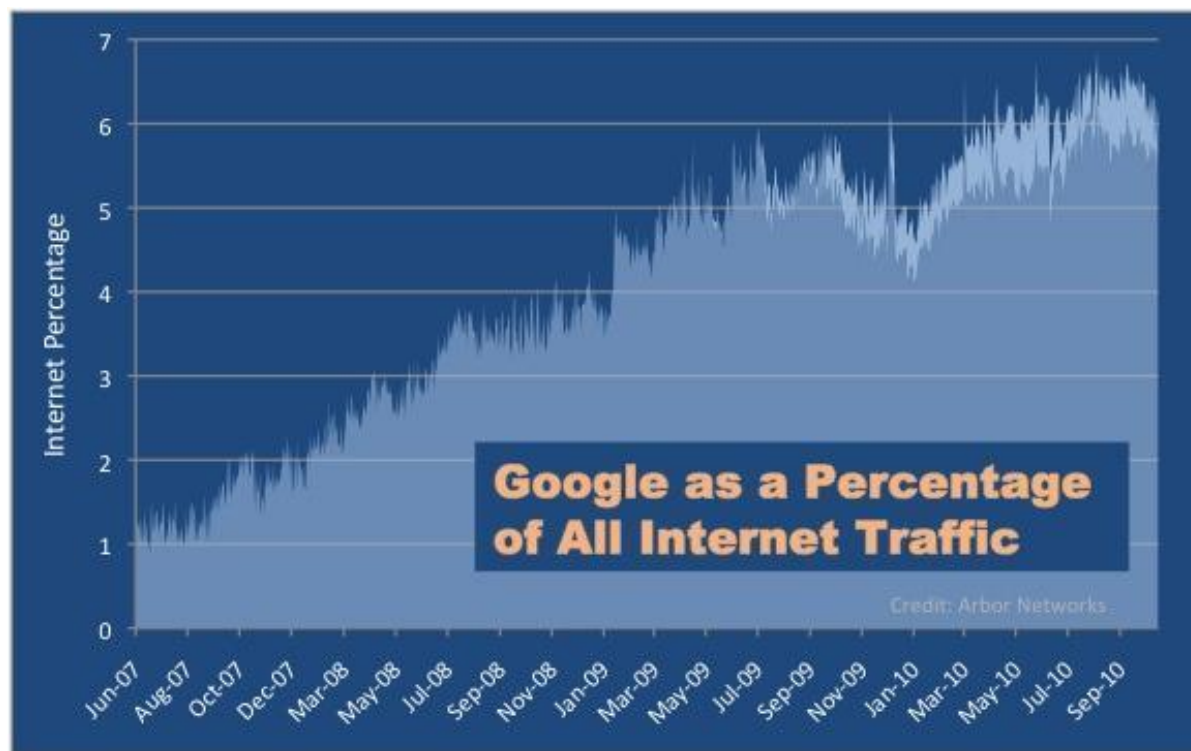
- Google operates two large backbone networks
 - Internet-facing backbone (user traffic)
 - Datacenter backbone (internal traffic)
 - Managing large backbones is hard
 - OpenFlow has helped us improve backbone performance and reduce backbone complexity and cost
 - I'll tell you how
-

Backbone Scale



“If Google were an ISP, as of this month it would rank as the second largest carrier on the planet.”

[ATLAS 2010 Traffic Report, Arbor Networks]



WAN-Intensive Applications



- YouTube
 - Web Search
 - Google+
 - Photos and Hangouts
 - Maps
 - AppEngine
 - Android and Chrome updates
-

- Cost per bit/sec delivered should go down with additional scale, not up
 - Consider analogies with compute and storage
- However, *cost/bit doesn't naturally decrease with size*
 - Quadratic complexity in pairwise interactions and broadcast overhead of all-to-all communication requires more expensive equipment
 - Manual management and configuration of individual elements
 - Complexity of automated configuration to deal with non-standard vendor configuration APIs

Solution: WAN Fabrics



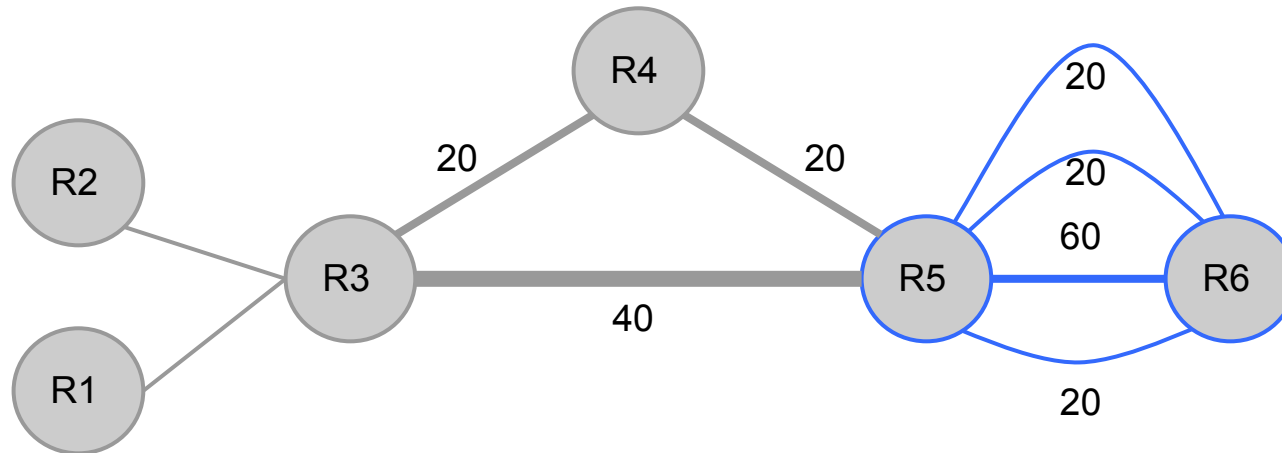
- Goal: manage the WAN as a *fabric* not as a collection of individual boxes
 - Current equipment and protocols don't allow this
 - Internet protocols are box centric, not fabric centric
 - Little support for monitoring and operations
 - Optimized for “eventual consistency” in routing
 - Little baseline support for low latency routing and fast failover
-

Motivating Examples

Convergence After Failure



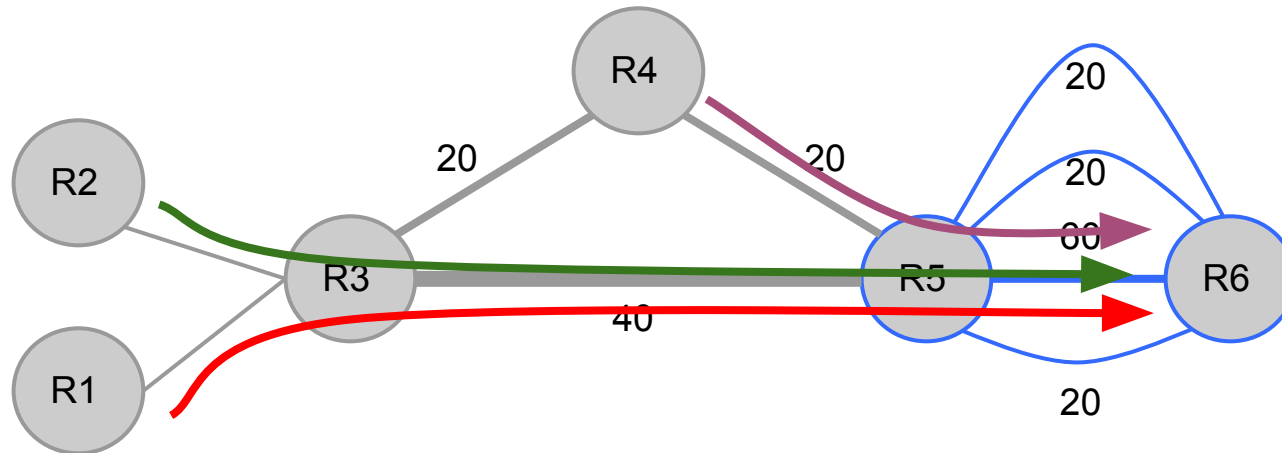
- Flows: R1->R6: 20; R2->R6: 20; R4->R6: 20



Convergence After Failure



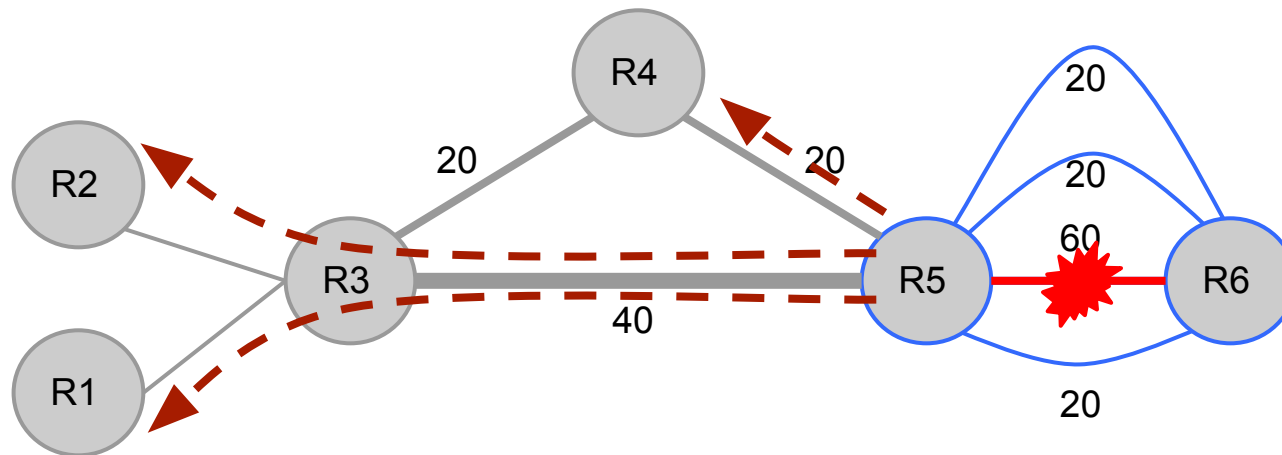
- Flows: R1->R6: 20; R2->R6: 20; R4->R6: 20



Convergence After Failure



- Flows: R1->R6: 20; R2->R6: 20; R4->R6: 20

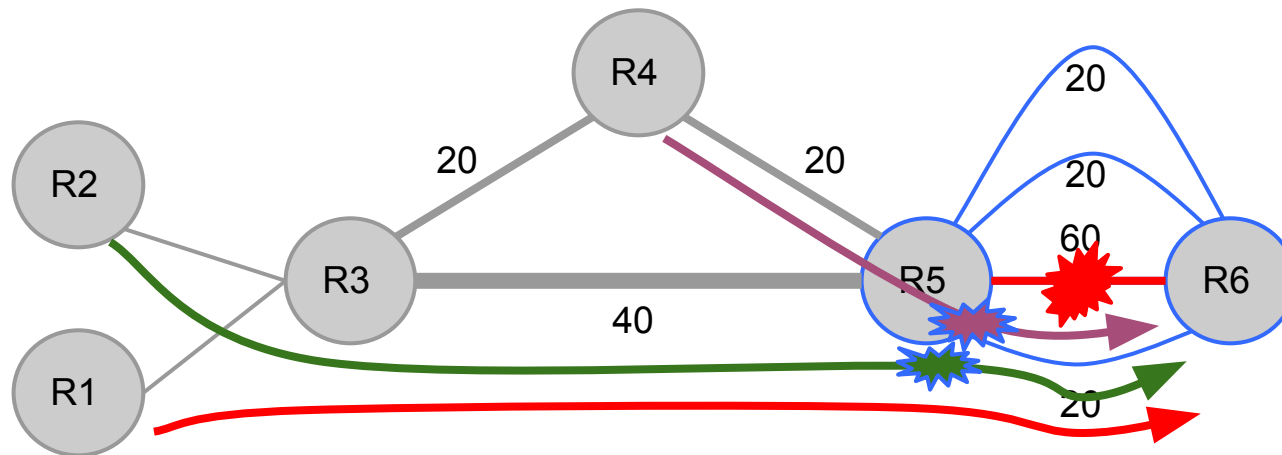


- R5-R6 link fails
 - R1, R2, R4 *autonomously* try for next best path

Convergence After Failure



- Flows: R1->R6: 20; R2->R6: 20; R4->R6: 20

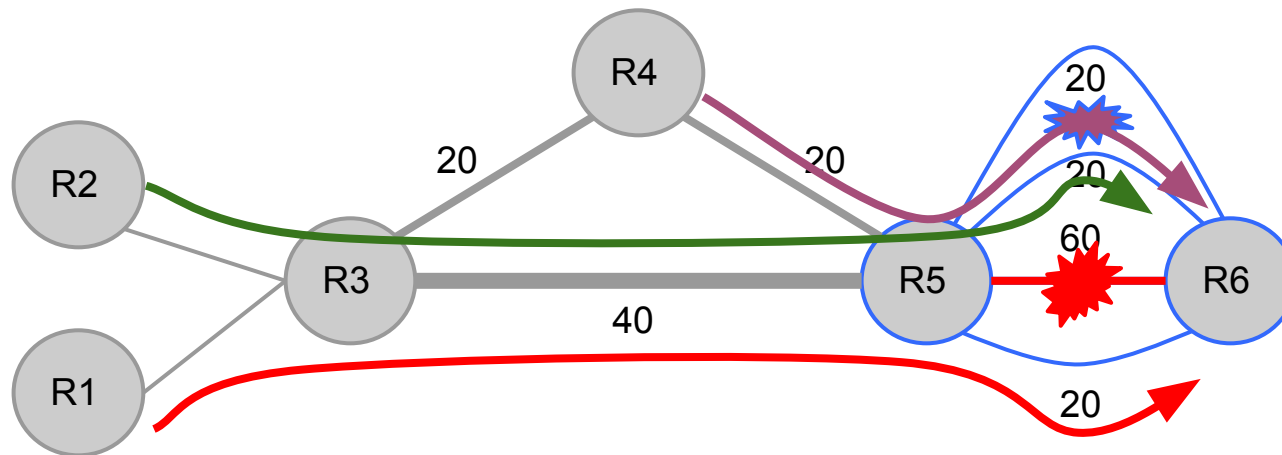


- R5-R6 link fails
 - R1, R2, R4 *autonomously* try for next best path
 - R1 wins, R2, R4 retry for next best path

Convergence After Failure



- Flows: R1->R6: 20; R2->R6: 20; R4->R6: 20

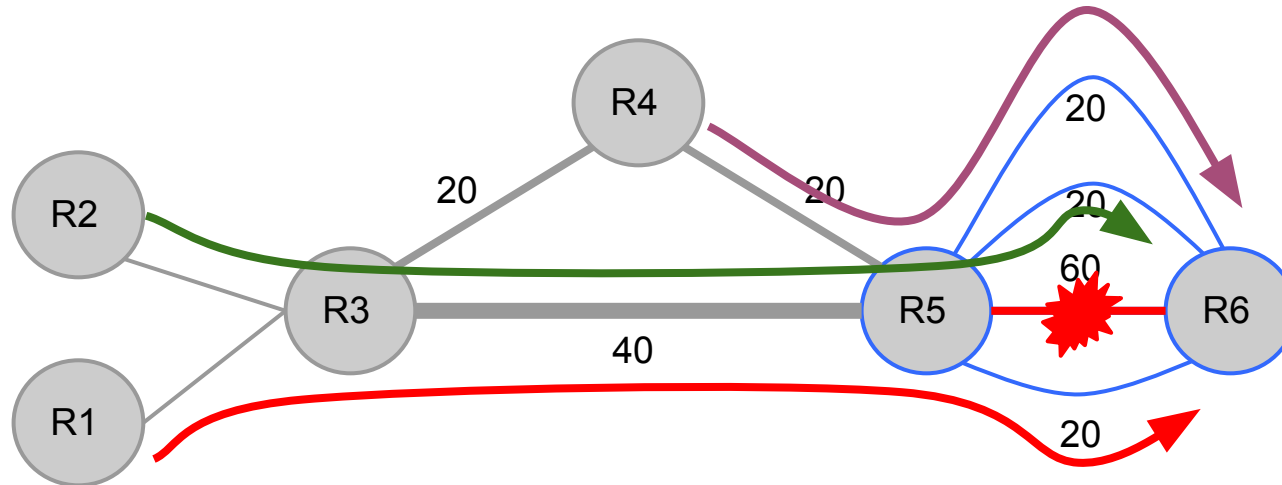


- R5-R6 link fails
 - R1, R2, R4 *autonomously* try for next best path
 - R1 wins, R2, R4 retry for next best path
 - R2 wins this round, R4 retries again

Convergence After Failure



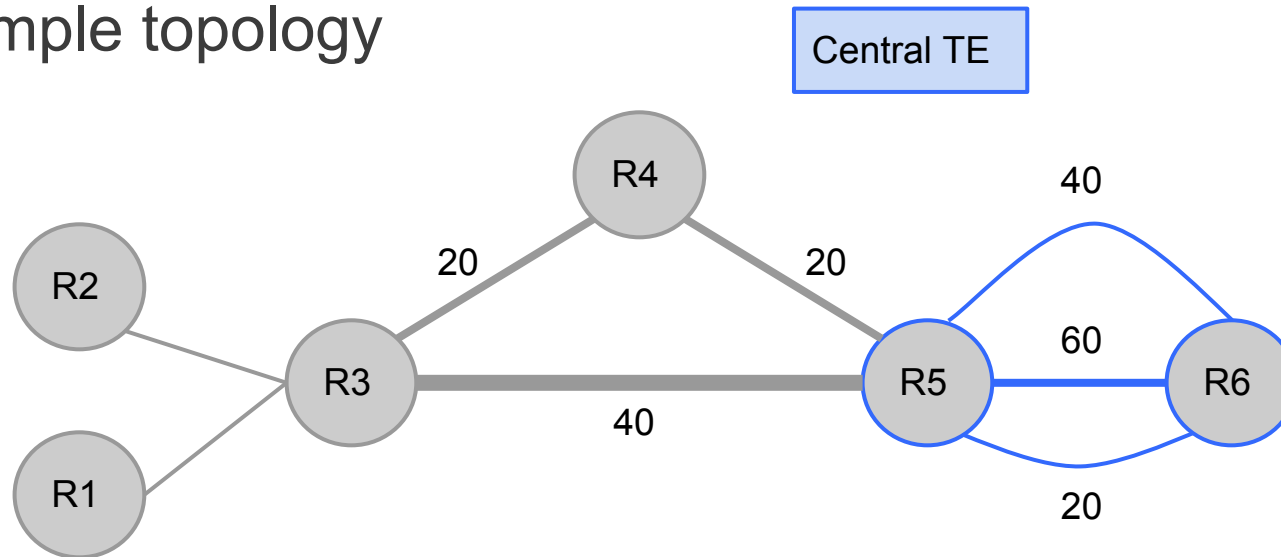
- Flows: R1->R6: 20; R2->R6: 20; R4->R6: 20



- R5-R6 link fails
 - R1, R2, R4 *autonomously* try for next best path
 - R1 wins, R2, R4 retry for next best path
 - R2 wins this round, R4 retries again
 - R4 finally gets third best path

Centralized Traffic Engineering Google™

- Simple topology

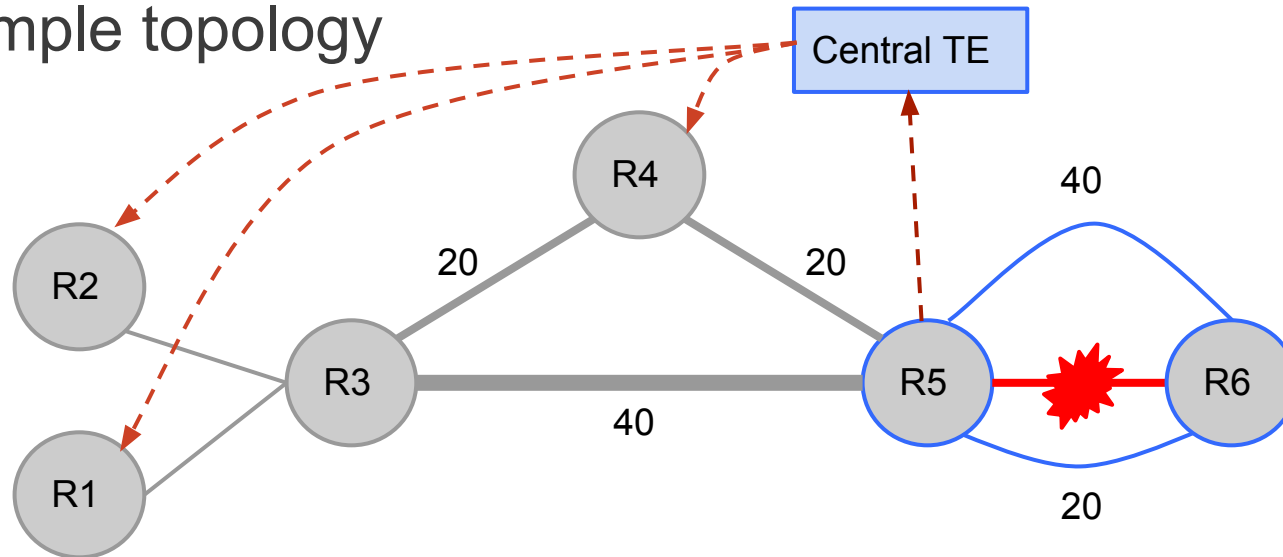


- Flows:

- R1->R6: 20; R2->R6: 20; R4->R6: 20

Centralized Traffic Engineering Google™

- Simple topology



- Flows:

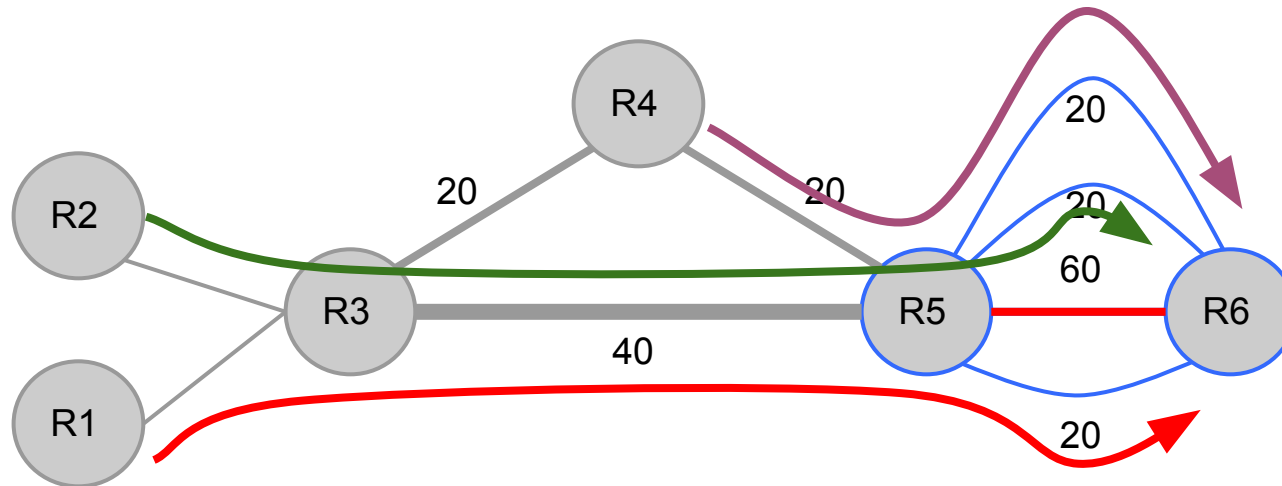
- R1->R6: 20; R2->R6: 20; R4->R6: 20

- R5-R6 fails

- R5 informs TE, which programs routers in one shot

Centralized Traffic Engineering Google™

- Simple topology



- Flows:

- R1->R6: 20; R2->R6: 20; R4->R6: 20

- R5-R6 link fails

- R5 informs TE, which programs routers in one shot
- Leads to faster realization of target optimum

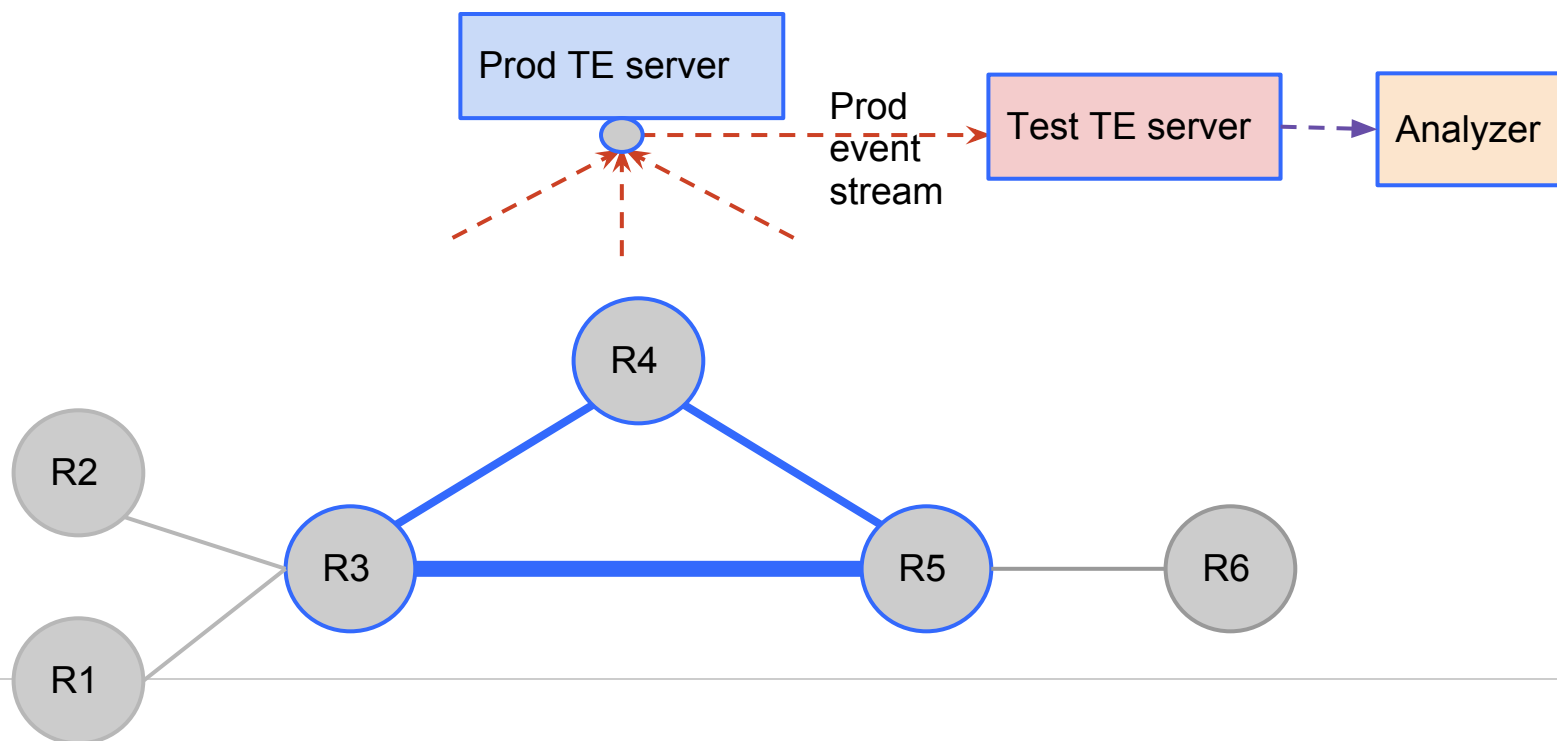
Advantages of Centralized TE

- Better network utilization with global picture
 - Converges faster to target optimum on failure
 - Allows more control and specifying intent
 - Deterministic behavior simplifies planning vs. overprovisioning for worst case variability
 - Can mirror production event streams for testing
 - Supports innovation and robust SW development
 - Controller uses modern server hardware
 - 50x (!) better performance
-

Testability Matters



- Decentralized requires a full scale replica of a real testbed to test new TE features.
- Centralized can tap real production input to research new ideas and to test new implementations



SDN Testing Strategy



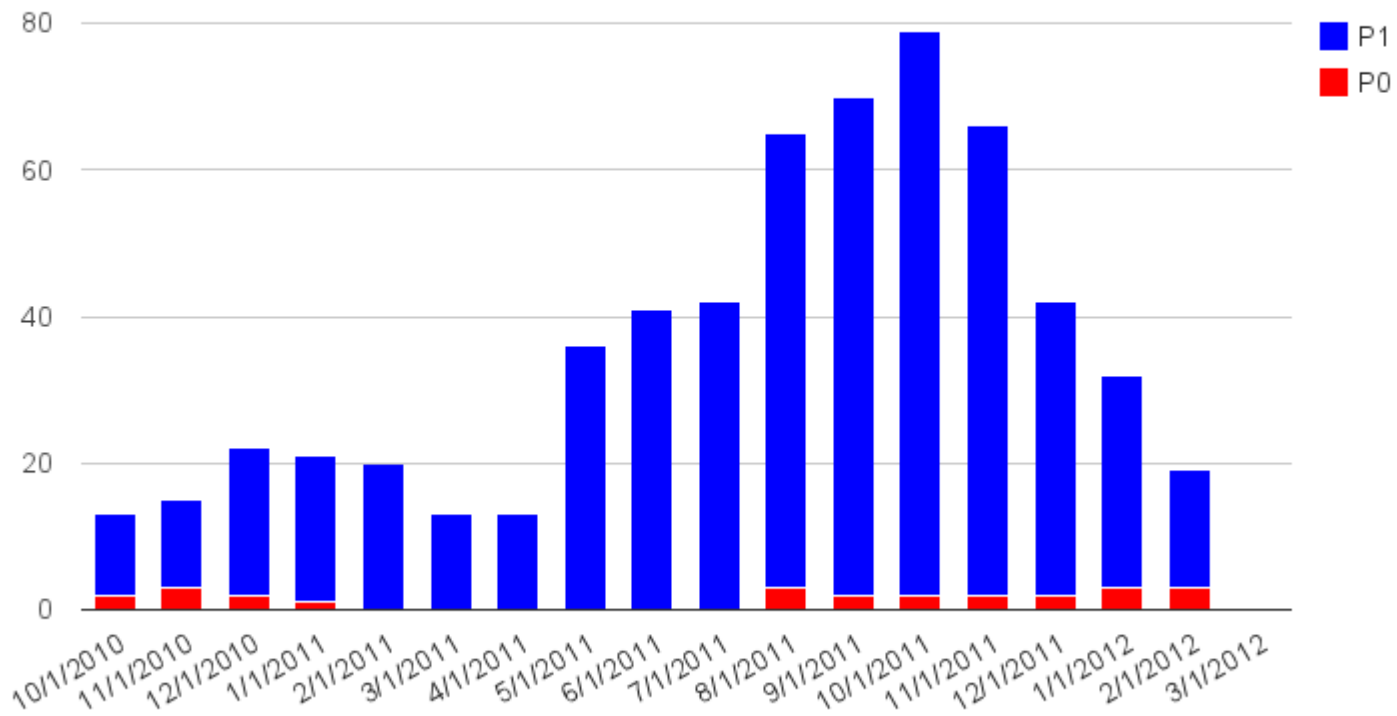
- Various logical modules enable testing in isolation
 - Virtual environment to experiment and test with the complete system end to end
 - Everything is real but hardware
 - Emphasis on in-built consistency checks (both during testing and in production)
 - Tools to validate programming state across all the devices. Validation checks can be done after every update from the central server (in virtual environment)
 - Enforce 'make-before-break' semantics
-

Our Simulated WAN



- Control servers run real binaries
 - Switches are virtualized
 - real OpenFlow binary, fake HAL
 - Arbitrary topology (can simulate entire backbone)
 - Can attach real monitoring and alerting servers
-

OFC Bug History



Why Software Defined WAN



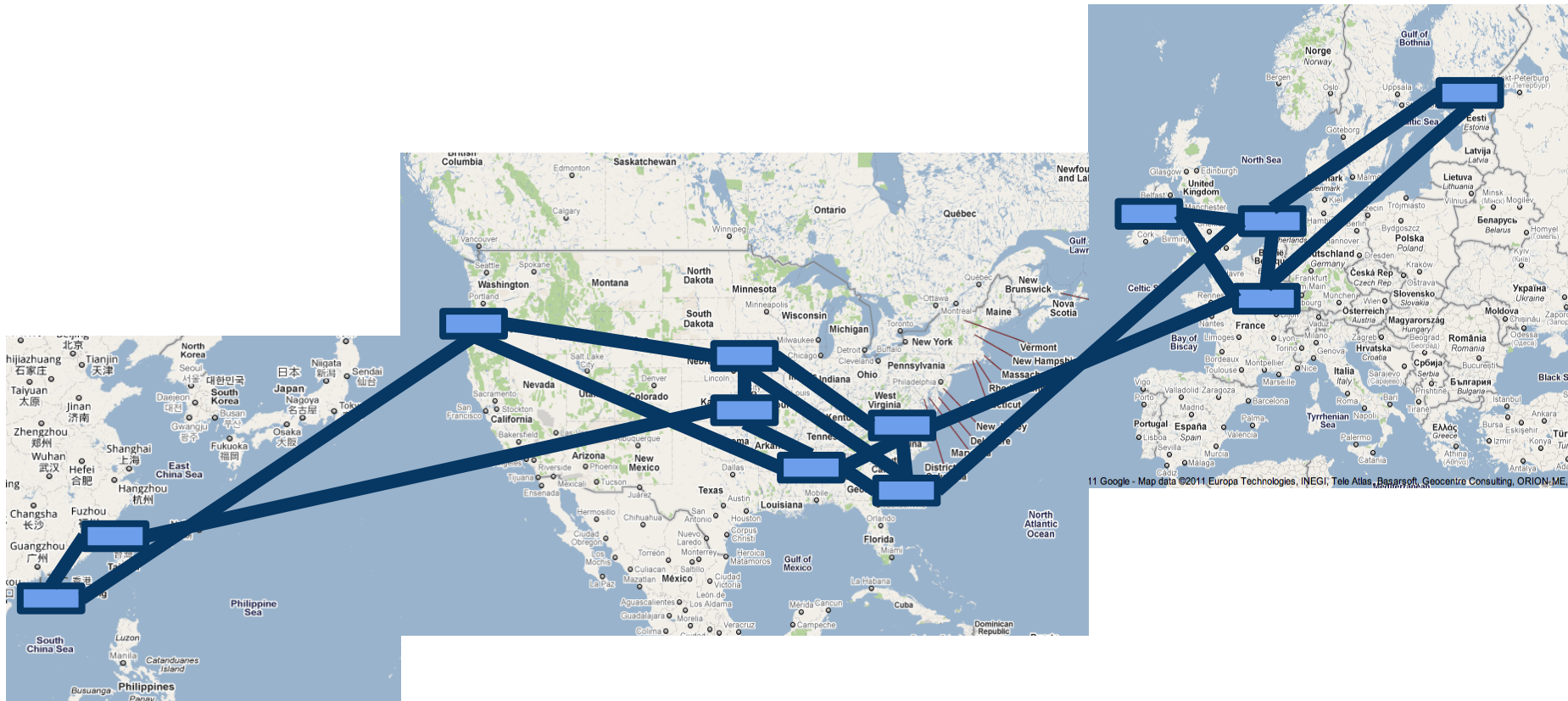
- Separate hardware from software
 - Choose hardware based on necessary features
 - Choose software based on protocol requirements
- Logically centralized network control
 - More deterministic
 - More efficient
 - More fault tolerant
- Separate monitoring, management, and operation from individual boxes
- *Flexibility and Innovation*

Result: A WAN that is higher performance, more fault tolerant, and cheaper

Google's OpenFlow WAN

- Two backbones
 - Internet facing (user traffic)
 - Datacenter traffic (internal)
- Widely varying requirements: loss sensitivity, availability, topology, etc.
- Widely varying traffic characteristics: smooth/diurnal vs. bursty/bulk
- Therefore: built two separate logical networks
 - I-Scale (bulletproof)
 - G-Scale (possible to experiment)

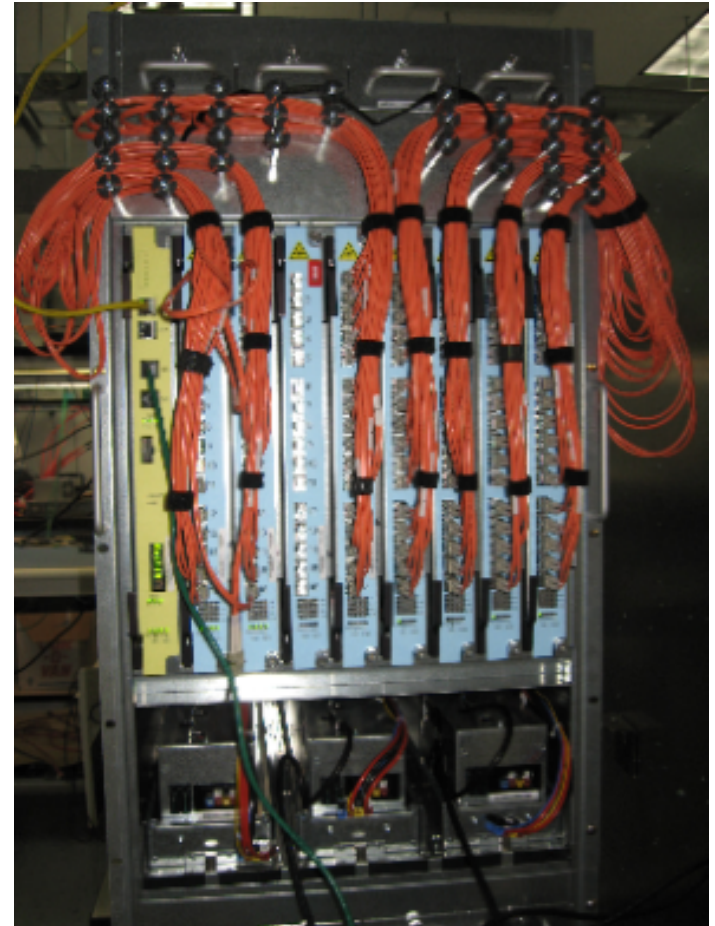
Google's OpenFlow WAN



G-Scale Network Hardware



- Built from merchant silicon
 - 100s of ports of nonblocking 10GE
- OpenFlow support
- Open source routing stacks for BGP, ISIS
- Does not have all features
 - No support for AppleTalk...
- Multiple chassis per site
 - Fault tolerance
 - Scale to multiple Tbps



G-Scale WAN Deployment



- Multiple switch chassis in each domain
 - Custom hardware running Linux
- Quagga BGP stack, ISIS/IBGP for internal connectivity

Deployment History



- Phase 1 (Spring 2010):
 - Introduce OpenFlow-controlled switches but make them look like regular routers
 - No change from perspective of non-OpenFlow switches
 - BGP/ISIS/OSPF now interfaces with OpenFlow controller to program switch state
 - Pre-deploy gear at one site, take down 50% of site bandwidth, perform upgrade, bring up with OpenFlow, test, repeat for other 50%
 - Repeat at other sites
-

Deployment History



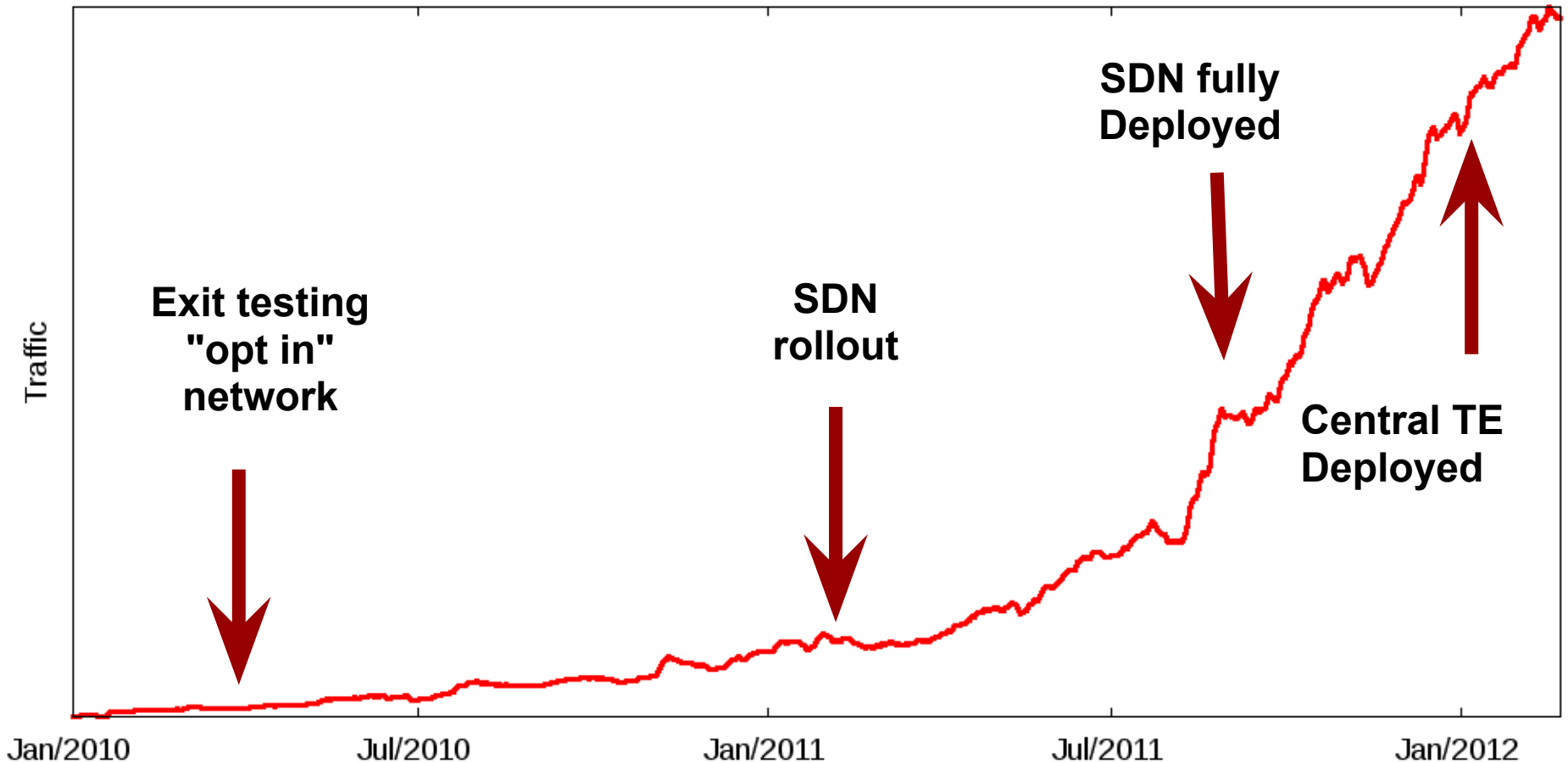
- Phase 2 (until mid-2011): ramp-up
 - Activate simple SDN (no TE)
 - Move more and more traffic to test new network
 - Test transparent roll-out of controller updates
-

Deployment History



- Phase 3 (early 2012): full production at one site
 - All datacenter backbone traffic carried by new network
 - Rolled out centralized TE
 - Optimized routing based on application-level priorities (currently 7)
 - Globally optimized placement of flows
 - External copy scheduler interacts with OpenFlow controller to implement deadline scheduling for large data copies
-

G-Scale WAN Usage



Google SDN Experiences



- Much faster iteration time: deployed production-grade centralized traffic engineering in two months
 - fewer devices to update
 - much better testing ahead of rollout
 - Simplified, high fidelity test environment
 - Can emulate entire backbone in software
 - Hitless SW upgrades and new features
 - No packet loss and no capacity degradation
 - Most feature releases do not touch the switch
-

Google SDN Experiences



- Already seeing higher network utilization
 - Flexible management of end-to-end paths for maintenance
 - Deterministic network planning
 - Generally high degree of stability
 - One outage from software bug
 - One outage triggered by bad config push
 - Too early to quantify the benefits
 - Still learning
-

Confirmed SDN Opportunities

- Unified view of the network fabric
 - Traffic engineering
 - Higher QoS awareness and predictability
 - Latency, loss, bandwidth, deadline sensitivity
 - Application differentiation
 - Improved routing
 - Based on a priori knowledge of the topology
 - Based on a priori knowledge of L1 and L3 connectivity
 - Improved monitoring and alerts
-

SDN Challenges



- Infancy of OF protocol
 - Still barebones but good enough
- Master election/control plane partition is challenging to handle
- What to keep on box what to remove? Not a perfect science
 - For things that remain on the box, how to configure?
- Flow programming can be slow for large networks
- All of the above are surmountable

Conclusions



- OpenFlow is ready for real-world use
 - SDN is ready for real-world use
 - Enables rapid rich feature deployment
 - Simplifies network management
 - Google's datacenter WAN successfully runs on OpenFlow
 - Largest production network at Google
 - Improved manageability
 - Improved cost (too early to have exact numbers)
-

Thank You!