Lessons Learned

(aka what’s transpired in these halls, but wasn’t intuitively obvious the first time)
Agenda

• Overview/Background
• POP architecture
• IGP design and pitfalls
• BGP design and pitfalls
• MPLS TE design and pitfalls
• Monitoring pointers
• Next steps
Overview

• Pete Templin, pete.templin@texlink.com
  – ‘Chief Card Slinger’ for a telecom/ISP
  – Hybrid engineering/ops position
• Recently acquired, now “strictly” engineering.
  – IP Engineer for a telecom/ISP
Objective: Simplicity

• “Be realistic about the complexity-opex tradeoff.” *Dave Meyer*

• Be realistic about the complexity, period.
  – Simple suggests troubleshootable.
  – Simple suggests scalable.
  – Simple suggests you can take vacation.
Be the router.

• When engineering a network, remember to think like a router.

• When troubleshooting a problem, remember to think like a router.
  – Think packet processing sequence, forwarding lookup method, etc. on THIS router.

• Work your way through the network.
  – Router by router.
Background

• {dayjob} grew from four routers (one per POP), DS3 backbone, and 5Mbps Internet traffic in 2003…
• …to 35 routers (4 POPs and a carrier hotel presence), NxDS3 backbone, and 200Mbps Internet in 2006…
• …and another 50Mbps since then.
When I started...

- ...I inherited a four-city network
  - Total internet connectivity was 4xT1
  - Static routes to/from the Internet
  - Static routes within the network
  - Scary NAT process for corporate offices
Initial challenges

• Riverstone routers – unknown to everyone
• Quickly found flows-per-second limits of our processors and cards
• We planned city-by-city upgrades, using the concepts to follow.
Starting point

• Everything starts with one router.
• You might run out of slots/ports.
• You might run out of memory.
• You might run out of processor(s).
• Whatever is your limiting factor, it’s then time to plan your upgrade.
Hardware complexity

• Once you grow beyond a single router, you’ll likely find that you need to become an expert in each platform you use.
  – Plan for this learning curve.
  – Treat product sub-lines separately
    • VIP2 vs. VIP4 in 7500s
    • GSR Engine revisions
    • Cat6 linecards (still learning here…)
Redundancy

- Everyone wants to hear that you have a redundant network.
- Multiple routers doesn’t ensure redundancy – proper design with those routers will help.
- If you hook router2 to router1, router2 is completely dependent on router1.
Initial design

• Two-tier model
  – Core tier handled intercity, upstream
    • Two core routers per POP
  – Distribution tier handled customer connections
    • Distinct routers suited for particular connections:
      – Fractional and full T1s
      – DS3 and higher WAN technologies
      – Ethernet services
Initial Core Design

• Two parallel LANs per POP to tie things together.
  – Two Ethernet switches
  – Each core router connects to both LANs
  – Each dist router connects to both LANs
Two core L2 switches
Pitfalls of two core L2 switches

• Convergence issues:
  – R1 doesn’t know that R2 lost a link until timers expire – multiaccess topology.

• Capacity issues:
  – Transmitting routers aren’t aware of receiving routers’ bottlenecks

• Troubleshooting issues:
  – What’s the path from R1 to R2?
Removal of L2 switches

• In conjunction with hardware upgrades, we transitioned our topology:
  – Core routers connect to each other
    • Parallel links, card-independent.
  – Core routers connect to each dist router
    • Logically point-to-point links, even though many were Ethernet.
Two core routers
Results of topology change

• Core routers know the link state to every other router.
  – Other routers know link state to the core, and that’s all they need to know.
• Routing became more predictable.
• Queueing became more predictable.
Core/Edge separation

• Originally, our core routers carried our upstream connections.

• Bad news:
  – IOS BGP PSA rule 9: “Prefer the external BGP (eBGP) path over the iBGP path.”
  – Inter-POP traffic left by the logically closest link unless another link was drastically better.
Lack of Core/Edge separation
Lack of Core/Edge separation

• Traffic inbound from city 2 wanted to leave via core1’s upstream, since it was an eBGP path.
  – City2 might have chosen a best path from core2’s upstream, but since each router makes a new routing decision, core1 sends it out its upstream.
Lack of Core/Edge separation
Problem analysis

• City1 core1 prefers most paths out its upstream, since it’s an external path.
• City1 core2 prefers most paths out its upstream, since it’s an external path.
• City2 core routers learn both paths via BGP.
• City2 core routers select best path as City1 core2, for one reason or another.
Problem analysis

- City2 sends packets destined for Internet towards City1 core1.
  - BGP had selected City1 core2’s upstream
  - IGP next-hop towards C1c2 was C1c1.
- Packets arrive on City1 core1
- City1 core1 performs IP routing lookup on packet, finds best path as its upstream link.
Lack of Core/Edge separation
Problem resolution

• Kept two-layer hierarchy, but split distribution tier into two types:
  – Distribution routers continued to handle customer connections.
  – Edge routers began handling upstream connections.
Core/Edge separation
Resulting topology

• Two core routers connect to each other
  – Preferably over two card-independent links
• Split downstream and upstream roles:
  – Downstream connectivity on “distribution” routers
    • Each dist router connects to both core routers.
  – Upstream connectivity on “edge” routers
    • Each edge router connects to both core routers.
Alternate resolution

- **MPLS backbone**
  - Ingress distribution router performs IP lookup, finds best egress router/path, applies label corresponding to that egress point.
  - Intermediate core router(s) forward packet based on label, unaware of destination IP address.
  - Egress router handles as normal.
IGP Selection

- Choices: RIPv2, OSPF, ISIS, EIGRP
- Ruled out RIPv2
- Ruled out EIGRP (Cisco proprietary)
- That left OSPF and ISIS
  - Timeframe and (my) experience led us to OSPF
  - Static routed until IGP completed!
IGP Selection

• We switched to ISIS for three supposed benefits:
  – Stability
  – Protection (no CLNS from outside)
  – Isolation (different IGP than MPLS VPNs)

• And have now switched back to OSPF
  – IPv6 was easier, for us, with OSPF
IGP design

• Keep your IGP lean:
  – Device loopbacks
  – Inter-device links
  – Nothing more

• Everything else in BGP
  – Made for thousands of routes
  – Administrative control, filtering
IGP metric design

• Credit to Vijay Gill and the ATDN team…
• We started with their model (OSPF-ISIS migration) and found tremendous simplicity in it.
• Began with a table of metrics by link rate.
• Add a modifier depending on link role.
## Metric table

- 1 for OC768/XLE
- 2 for OC192/XE
- 3 for OC48
- 4 for GE
- 5 for OC12
- **We’ll deal with CE, CLXE, and/or OC-3072 later!**
- 6 for OC3
- 7 for FE
- 8 for DS3
- 9 for Ethernet
- 10 for DS1
Metric modifiers

• Core-core links are metric=1 regardless of link.
• Core-dist links are $500 + \langle\text{table value}\rangle$.
• Core-edge links are $500 + \langle\text{table value}\rangle$.
• WAN links are $30 + \langle\text{table value}\rangle$.
• Minor tweaks for BGP tuning purposes.
  – Watch equidistant multipath risks!
Metric tweaks

• Link undergoing maintenance: 10000 + <normal value>
• Link out of service: 20000 + <normal value>
• Both tweaks preserve the native metric
  – Even if we’ve deviated, it’s easy to restore
Benefits of metric design

• Highly predictable traffic flow
  – Under normal conditions
  – Under abnormal conditions

• I highly recommend an awareness of the shortest-path algorithm:
  – Traffic Engineering with MPLS, Cisco Press
  – My NANOG37 tutorial (see above book…)
Metric design and link failure

• Distribution/edge routers aren’t sized to handle transitory traffic.
• Distribution/edge routers might not have proper transit features enabled/configured.
• If the intra-pop core-core link(s) fail:
  – We want to route around the WAN to stay at the core layer.
Metric design and link failure

• Core-dist-core or core-edge-core cost:
  – At least 1002 (501 core-dist and 501 dist-core)

• Core-WAN-core cost:
  – At least 63 (31 core-cityX, 1 core-core, 31 cityX-core)
  – Additional 32-40 per city

• Traffic would rather traverse 23 cities than go through distribution layer.
IGP metric sample
Pitfalls of metric structure

• Links to AS2914 in Dallas, Houston
  – Remember IOS BGP PSA rule 10: “Prefer the route that can be reached through the closest IGP neighbor (the lowest IGP metric).”
  – SA Core1 was connected to Dallas
    • Preferred AS2914 via Dallas
  – SA Core2 was connected to Houston
    • Preferred AS2914 via Houston
Pitfalls of metric structure

- Dallas was sending some outbound traffic to AS2914/Houston because of IGP metric.
- Houston Edge1 metrics were changed to rebalance traffic.
- SA dist routers had BGP multipath enabled.
- Four dist routers ran out of RAM simultaneously.
BGP design

- BGP is made to scale: use it
  - Customer link subnets
  - Customer LAN subnets
  - External routes

- BGP has great filtering tools: use them
  - Filter at every ingress and route injection point
  - Apply an internal community
BGP scaling pitfalls

• Confederations didn’t work well for us
  – One sub-AS per POP meant each router was its own sub-AS.
  – Convergence was painful; sub AS path tried to be an IGP.

• Removed confederations then deployed route reflectors
  – No client-client reflection for easier scaling.
BGP at distribution layer

• Redistribute connected routes into BGP
  – Exclude the interfaces already handled in IGP
    • Oops: don’t write your route map to exclude by interface name. One failed VIP or LC now causes a deny-all
    • Instead, exclude your IGP interfaces by prefix list.

• Redistribute static routes into BGP

• No customer configurations are needed anywhere else
BGP local-pref design

- Transit: cost $ money
- Peering: usually low or no cost
- Customers: revenue
- Treat prefixes appropriate to dollars
  - Prefer to send to customer rather than through peering or transit
  - Often used: local preference
Local preference design

- Customer LP = 400
- Peer LP = 300
- Transit LP = 200
- Backup LP = 50
- Since default LP is 100, a forgotten or flawed route map will result in routes that aren’t used.
  - The error will become apparent!
Customer filtering plan

• Filter once on ingress
• Do so **aggressively:**
  – We filter on \{prefix, AS-path\}
  – We allow customer to prepend freely
  – We allow customer to truncate the AS-path
    • Second and subsequent AS is optional
  – We tell customer about filtering rules (and lots more) at turn-up.
Customer route filtering, part 1

- Accept null-routed aggregate
  - Set next-hop for null
  - Propagate normally

- Accept aggregate
  - Propagate normally
Customer more-specifics filter

• Accept null-routed specific
  – Set next-hop for null, mark as no-export
  – Propagate internally

• Accept specific w/ ‘override’ community
  – Treats as aggregate (propagated out)
  – Hopes transits filter on ‘le 24’
  – Best-effort option
Customer more-specifics, cont.

• Accept specific
  – Mark as no-export
  – Propagate internally
  – Used as uRPF opening for traffic engineering
Customer filtering logic

- Customer can announce aggregate.
- Customer can announce aggregate with null-routed specifics.
- Customer can announce aggregate AND null-route it, announce more-specifics to forward.
  - And can null-route further specifics.
Customer filtering sample

- 72.18.90.0/22 with 11457:0
  - Aggregate is null-routed, but is announced to the world.
- 72.18.92.0/23
  - More-specific is shared within AS, traffic is forwarded to customer
- 72.18.93.0/24 with 11457:0
  - More-specific is null-routed.
- Only 72.18.92.0/24 is forwarded to customer.
Impact of filtering

- We have at least two prefix lists per customer:
  - One exact-match list per allowed AS path
  - One ‘le 32’ list for null routing and overrides
- We can optionally inject ‘tuning communities’ in the customer inbound route-map
BGP community design

• Tag every prefix with an internal community at ingress.
  – Identify POP of origin
  – Identify requested egress handling
  – Identify type of route (customer, ours, external)

• Use the tag intelligently:
  – Use the POP of origin to adjust MED
    • “Simple” geo-routing for customer prefixes saved us significant WAN costs.
Our internal community design

• 11457:ABCDE
  – A is route type (1=cust, 2=ours, 3=upstream, etc.)
  – BC is POP of origin
  – D is desired tuning (0=as-tuned, 1=provider-default, 2=backup, 7=maintenance)
  – E is georouting (0=aggregate, hot potato, 1=POP-specific, cold potato)
Internal community, sample

• 11457:10200
  – A=1, so it’s a customer route
  – BC=02, so it came from POP#2 (Dallas)
  – D=0, so we propagate based on default tuning (possibly prepends and/or localpref tweaks)
  – E=0, so we announce as hot-potato (equal default MED in all cities)
Georouting

- Each provider port has a community list that matches “nearby” POPs.
  - If internal community matches 11457:.....1 and nearby POPs, MED=200.
  - If internal community matches 11457:.....1 but not nearby POPs, MED=400.
  - If internal community matches 11457:.....0, MED=200.
BGP community design

• Develop a set of communities that you or your customers can apply to routes for tuning within your network:
  – Set local preference
  – Null route

• Customers can create cust/cust-backup or peer/peer-backup by using MED and LP.
Our customer community design

• 11457:localpref
  – For limited versions of localpref (200, 300, 400)

• 11457:0
  – For null routing
BGP tuning design

• Develop another set of communities that you or your customers can apply to routes for tuning outside your network:
  – No-advertise
  – Set prepends
  – Request local preference
Announcement tuning logic

Filter out other upstream routes

Allow routes flagged with individual or global LP/prepend requests - complex to handle combos

Allow routes flagged with internal LP requests and map a corresponding LP

Process routes based on embedded tuning (11457:ABCDE)

Set MED based on embedded tuning
BGP outbound tuning

• We “enjoy” parallel connectivity to three transit providers
  – For each, one link in Dallas, one link in Houston.
• Cold potato to transit providers’ space and their customers
• Hot potato beyond their network
BGP outbound logic

• In normal state, cold potato is only one hop longer than hot potato for us.
  – We know our network
  – They know their network
  – But, we know our network better than we know their network.
  – If they’re telling us a particular POP is better, we’ll use it.
BGP outbound logic

• Assumption is MED learned reflects IGP distance to point of (aggregate) injection.
  – For transit providers’ routes, point us towards the point of aggregate origination.
  – For transit provider’s customers, since MED won’t traverse directly, assume provider has chosen a best path (based either on customer MED or hot/cold potato) and MED leads us there.
Customer BGP experience

• We respect that many (all?) of our customers have little to no BGP experience.
• As long as customer sends their aggregate with a reasonable AS path and not too many routes to bump against max-prefix, OK.
• We’ll apply reasonable tweaks at customer request, but otherwise let them know they have all the knobs they’ll need.
Traffic Engineering

• Redundancy is hard to plan
  – Do you conduct regular simulations?
  – Some networks aren’t conducive to efficient redundancy.

• “Two means one, one means none”
  – From the movie “GI Jane”

• 2/1 means half of your capacity is excess.
  – Ugh.
MPLS Traffic Engineering

• MPLS TE saved our network
  – Normal IGP/EGP routing is completely unaware of traffic saturation, until enough keepalives are lost.
  – MPLS TE enables routers to spread traffic over multiple paths, including those that are not the shortest IGP path.
  – Built using one-way tunnels between routers.
MPLS TE deployment

• Initial deployment:
  – Full mesh of tunnels between dist and edge routers, with 1-2 tunnels depending on traffic loads.
  – Aggressive (15-minute) auto-bandwidth timers meant that the network was adjusting rapidly.
  – Our backbone, versus the size of the major flows, required this approach.
MPLS TE pitfalls

• NNTP: few large-bandwidth flows would get glued to a tunnel.
  – Add tunnels for granularity.

• Redundant capacity can easily get used by accident – no easy tracking.
  – However, excess capacity can get used during momentary surprises!
MPLS TE long-term

• IOS issues eventually caught us
  – End solution is entirely within the core layer, and only across WAN links.
  – Standard deployment of four tunnels per link.
  – Roughly 25% of traffic swings at a time
  – Traffic follows lowest-metric topology except during congestion.
Monitoring

• Consider home-grown tools to research many/all facets of a particular customer’s port/service
  – Consolidate relevant information for your help desk
  – Minimize the need to share ‘enable’
Monitoring

• Three problems to solve:
  – What is up/down at this moment?
  – What happened when?
  – How many [bits, packets, errors, etc.] are flowing?

• Usually different tools to solve each problem.
Monitoring

• For us, the two biggest things were MRTG with home-brew enhancements and syslog.
  – Our MRTG has simple links per port for a cutesy network diagram, telnet to CPE, and how-to-configure a CPE
  – Our syslog has a Perl wrapper that color-codes up/down and substitutes in the interface description so the entry has local meaning.
Sample diagram

Network Diagram for Wall Homes

Local WAN Address (IAD Far End) 66.118.5.149
WAN Network 66.118.5.148/30
Remote WAN Address (IAD Local) 66.118.5.150

(T1)
(dist 3/0 T1#7)
timeslots 1-15

(Customer Network)

LAN Network (Subnet Masks) 66.118.21.208/28
Likely LAN Gateway 66.118.21.209
Admin Internal Distance Tag
Sample log watcher

Jan 22 13:03:16 dist6-dlls 976: Jan 22 19:03:11.645 GMT: %LINK-3-UPDOWN: Serial2/1/4:1 (PM Realty - Dallas Parkway), changed state to down
Jan 22 13:03:16 dist6-dlls 977: Jan 22 19:03:12.645 GMT: %LINEPROTO-5-UPDOWN: Line protocol on Serial2/1/4:1 (PM Realty - Dallas Parkway), changed state to down
Jan 22 13:03:48 dist6-dlls 978: Jan 22 19:03:44.233 GMT: %CONTROLLER-5-UPDOWN: Controller T3 2/1 T1 4, changed state to UP
Jan 22 13:03:50 dist6-dlls 979: Jan 22 19:03:45.945 GMT: %LINK-3-UPDOWN: Serial2/1/4:1 (PM Realty - Dallas Parkway), changed state to up
Jan 22 13:03:51 dist6-dlls 980: Jan 22 19:03:46.949 GMT: %LINEPROTO-5-UPDOWN: Line protocol on Serial2/1/4:1 (PM Realty - Dallas Parkway), changed state to up
Jan 22 13:05:32 dist6-dlls 981: Jan 22 19:05:27.669 GMT: %CONTROLLER-5-UPDOWN: Controller T3 2/1 T1 4, changed state to DOWN
Jan 22 13:05:34 dist6-dlls 982: Jan 22 19:05:29.668 GMT: %LINK-3-UPDOWN: Serial2/1/4:1 (PM Realty - Dallas Parkway), changed state to down
Jan 22 13:05:34 dist6-dlls 983: Jan 22 19:05:30.668 GMT: %LINEPROTO-5-UPDOWN: Line protocol on Serial2/1/4:1 (PM Realty - Dallas Parkway), changed state to down
Jan 22 13:06:03 dist6-dlls 984: Jan 22 19:05:59.036 GMT: %CONTROLLER-5-UPDOWN: Controller T3 2/1 T1 4, changed state to UP
Jan 22 13:06:05 dist6-dlls 985: Jan 22 19:06:00.816 GMT: %LINK-3-UPDOWN: Serial2/1/4:1 (PM Realty - Dallas Parkway), changed state to up
Jan 22 13:06:06 dist6-dlls 986: Jan 22 19:06:01.820 GMT: %LINEPROTO-5-UPDOWN: Line protocol on Serial2/1/4:1 (PM Realty - Dallas Parkway), changed state to up
Security

• Prevent bad traffic
  – BCP38 (anti-spoofing)
  – Use uRPF unless you can’t, please
  – Allows a simple but effective inbound ACL
    (less complexity in older GSR cards)

• Block it before it ever gets into your network!
Security

• Black hole routing
  – Cannibalize a 2511 as a black hole trigger
  – Google “RTBH”

• Build at least the most basic NetFlow infrastructure
  – Learn how to find DDOS (think “sort by packets in flow”) and black hole fast
Closing

• That’s my story, and I’m sticking to it.
  – It’s worked very well for us. My phone rings with a “stumper” every three months or so.

• Configuration snippets from any part of our network are available by email request.

• Questions?