

# Intelligent Routing for Success in Content Delivery Networks: The Juniper Networks JUNOS Advantage

*by Jeffrey Papen, Peak Web Consulting*

## Introduction

Content Service Providers (CSP) have unique data networking challenges compared to other networking environments, requiring an equally unique set of tools. Factors that differentiate the CSP's network from many other network environments are the risk of economic impact from misrouting multiple gigabits per second of transit traffic, the requirement for 100% uptime, and the complexity of configuration with distributed datacenter interconnections. A Content Service Provider's network is the lifeline to its customers and is therefore intolerant of disruptions or outages, which may damage the company's reputation and market position. With 10 Gb/s Ethernet links increasingly the norm, and pressures mounting in many networks for 40 and 100 Gb/s links with their associated higher infrastructure costs, network engineers can no longer mitigate risks simply with the common practice of over-engineering their networks, but rather must find creative solutions based on more intelligent routing versus brute force spending.

The choice of networking platform can offer CSPs real technical and financial advantages in terms of advanced functionality and superior performance. Juniper Networks' product portfolio provides packet-handling intelligence, ease of use in configurations, and embedded methods to prevent inadvertent errors, combined with best-in-class routing hardware and software.

This paper highlights the critical needs of CSPs that are best served by Juniper's platform and the JUNOS network operating system, and considers six major points regarding how Juniper delivers measurable advantages to a CSP network in ways not possible with other vendors:

- BGP implementation ease of use
- Reduced operator error
- Leveraging routing policies to achieve economic and business goals
- Better information and insight into your business
- Stability of platform with a ubiquitous code train

## BGP implementation ease-of-use

Multihoming (connecting to multiple ISPs) has become as necessary to CSP environments as the use of redundant power sources or multiple datacenters. No Content Business can afford prolonged outages, and the first and surprisingly most effective way to maximize uptime is through a sane and robust BGP implementation to multiple transit providers.

While BGP remains the most complex of routing protocols to understand fully and implement on a global scale, the number of engineers who are familiar with the protocol's intricacies, as well as the constantly changing Internet landscape, continue to increase. The tools unique to Juniper Networks reduce the learning curve for working with BGP considerably. Without diving into the details of the protocol, AS (autonomous system) numbers and route communities are two of the many aspects of the protocol that are painful to administer.



**Within JUNOS, mapping the numbers of long-forgotten peers, obscure customer or target networks, or infrequently used ISPs to names that are easy to remember is achieved with a single line of configuration.**

AS numbers are globally unique identifiers that associate a routing announcement with the company that owns that network. These are required when connecting to the Internet via multiple paths, and are used by a CSP to identify the networks from which consumers are connecting. There are currently over 27,000<sup>1</sup> active AS numbers on the public Internet, and with the impending conversion from 2-Byte to 4-Byte numbers, keeping them straight will be even more of a challenge for CSPs.

Relying on an engineer's memory of AS numbers during configurations, or even worse, during late-night outage troubleshooting, can often lead to errors. Furthermore, when familiarity with AS numbers is a requirement for effective network engineering, the barrier to entry for new staff is extremely steep, and senior-level engineers who could be better utilized focusing on strategic work are still relied upon for day-to-day BGP management tasks.

Juniper's JUNOS operating system makes managing BGP easier by mapping AS numbers to easily readable names. How many Internet Engineers can name all 28 ASNs that make-up the Comcast constellation of networks? I would imagine not even their own staff has every ASN memorized. Within JUNOS, mapping the numbers of long-forgotten peers, obscure customer or target networks, or infrequently used ISPs to names that are easy to remember is achieved with a single line of configuration. Additionally, mapping groups of ASNs into a single collection is also implemented with ease.

JUNOS also simplifies working with BGP route communities compared with other implementations. BGP communities are optional tags associated with a route that produce no changes to the routes themselves, but rather allow network engineers to easily identify routes with a common characteristic in order to apply consistent behavior to routes within the community. For example, an often-used policy is to tag all routes learned from upstream transit providers with a "transit" community tag, preventing these routes from being advertised to settlement-free peers. Taking this basic property even further, an additional benefit of communities is that you can add multiple communities to each route, providing an incredibly deep well of information for other networks to draw upon. Communities are often used to identify if the route is learned from a transit provider, a peer, or a customer, what part of the country the route is learned from, whether the route should be advertised upstream or downstream, or to direct an ISP to prefer or not prefer an advertised route. A BGP community is often written as a sixteen-bit number, then a colon, then another sixteen-bit number. The first sixteen-bit number is used to identify the autonomous system number (ASN) of the network adding the community tag. The second sixteen-bit number is used to characterize the route.

The difficulty for a CSP in utilizing communities learned from ISPs is that there is no universal standard for the second sixteen-bit number: each autonomous system establishes its own numbering scheme. For example, to identify a downstream "customer" route, Level 3 uses 3356:123<sup>2</sup>, while NTT uses 2914:410<sup>3</sup>. Many community definitions are published via different methods that will force a CSP network engineer to hunt around, whereas others are guarded as "proprietary" and kept private. If your network were multihomed to Level 3 and NTT and you're trying to decide how to divide traffic between the two networks, a good place to start would be to send traffic to each ISP's directly-attached customers first, assuming that the number of congestion points would be minimized by traveling through the fewest possible ISPs. Your policy would have to identify routes tagged with 3356:123 and 2914:410. These numbers are non-intuitive, and when new staff is brought up to speed, the ability to ascertain the intention of this BGP routing policy is lost in Google searches for each of these community definitions.

With Juniper Networks routers, engineers can set definitions such as Level 3:Customers or NTT:Customers, immediately detailing what types of routes the policy matches on, and subsequently modifies preference for.

---

<sup>1</sup> <http://bgp.potaroo.net/as1221/bgp-active.html>

<sup>2</sup> <http://lg.level3.net/whois/whois.cgi?server=rr.level3.net&query=as3356/>

<sup>3</sup> <http://www.us.ntt.net/about/policy/routing.cfm>

**JUNOS is the only operating system that provides staged, atomic updates to configurations, where changes do not take effect until explicitly committed in their entirety.**

Alternatively, when reading a policy designed to influence the inbound path for traffic, which would make the most sense: a policy that set some routes with the community 3356:70, and the rest with 3356:90, or a policy that set routes with the communities “Level3:Less-Preferred-Than-Peers” and the rest with “Level3:More-Preferred-Than-Peers-But-Lower-Than-Customer-Default?”. The power of communicating the intent of a community within the community name itself saves countless hours of troubleshooting, pilot error, and weeping/gnashing of teeth.

**Personal Experience #1.** As CSPs grow and their networking requirements become more complex, the need for descriptive communities is critical. Seven years ago while multihoming a major CSP’s network, our BGP policies for various ISPs could require ten or more conditional checks applied to each route to assign various local-pref, MEDs, or even origin. Without easily defined names, my fellow engineers would have required a Rosetta Stone to translate these policies and maintenance would become impossible.

When using communities to modify your advertisements outbound to upstream ISPs, community aliases take on an even more powerful role for communicating the desired outcome of your policy; in the heat of troubleshooting, this is an invaluable time saver that also greatly reduces the most common cause of outages: pilot error.

**Personal Experience #2.** When networks begin to peer<sup>4</sup>, the number of ASNs one interacts with jumps by at least one--if not two--orders of magnitude. Even when dealing with ISPs on a daily basis, there are countless smaller ISPs and single-location peers that are too difficult to remember. JUNOS’s basis within FreeBSD and native SSH Version 2.0 support made the distribution of ASN mappings trivial. We maintained a single file with all ASN to name mappings, and using shell scripts, could push this file to every Juniper router. Now whenever one router would peer with an ASN, if we were troubleshooting a problem, it was trivial on any other router within the network to search the config for that mystery ASN we didn’t have memorized.

## Reduced operator error

No matter how much we like to believe that network engineers and architects are all-knowing and perfect, the majority of all outages are caused by pilot error (human mistakes). While the reasons behind pilot error are as varied as the types of errors that can unfold, JUNOS provides more opportunities to almost eliminate outages by catching those mistakes before becoming catastrophic. Should pilot error cause a network disruption, JUNOS provides the easiest and fastest recovery method via a single command, or even automatically.

## Atomic commits

In non-Juniper platforms, each configuration change takes effect immediately, as typed on the operational command line. With no way to make a group of changes take place as a whole, one is often left to try pasting a bunch of commands and hoping against a race condition. Configurations entered in the wrong sequence can often lead to unintended effects, especially when policies and security restrictions are applied. JUNOS is the only operating system that provides staged, atomic updates to configurations, where changes do not take effect until explicitly committed in their entirety. Entering the changes one a time, with no rush, you can use commands such as **show | compare** (to show differences between the current and the proposed configuration), and **commit check** (which performs full semantic checking on the entire config without enabling the changes) to double- and triple-check your work. If any errors are found in these verification steps, all changes can easily be reverted with the simple **rollback** command. Then only when you’re sure – or when the maintenance window has arrived – would you make the configuration changes active with the **commit** command.

---

<sup>4</sup> Peering is a direct connection between two ISPs where only their address or possibly customer address space is advertised. The cost for this bandwidth is lower than the cost of transit to an ISP and often free. Peering’s goal is to lower costs, improve performance via direct connects, and to reduce the dependency on ISPs for global connectivity.

BGP sessions don't seem like an obvious candidate for race conditions, but if you prefer to type your BGP session by hand on a non-Juniper router, then watch out. The default behavior for BGP is to advertise and learn all valid BGP routes. Once a BGP session establishes, the only way to change what routes are advertised is by soft or hard clearing of the session. It only takes one line to establish a BGP session and that session may establish in just seconds. If the BGP session establishes before typing the commands limiting the routes to be learned and/or advertised, then the commands are silently ignored. Many times I have been typing away on a new BGP session that came up in the background before I added my outbound policy limiting the routes advertised to my neighbor to only my local routes, and I inadvertently became a transit provider to a peer. With JUNOS' atomic commit, engineers will be sure to have policies in place before BGP sessions can establish.

## Undoing bad configuration changes

Even the most careful engineer is going to commit the wrong change, sometimes with catastrophic results, but JUNOS provides a saving grace for this scenario as well. Adding the word **"confirm"** after the command **"commit"** makes the configuration live, but requires a second commit to make the change permanent. If you don't type that confirming **commit** within the timeout period, then JUNOS will revert back to the previous configuration, undoing all of your (potentially bad) changes. How many times have you locked yourself out of a router by applying a firewall to the interface you didn't realize was your way into the router? With **"commit confirmed"**, if you lock yourself out, JUNOS lets you back in.

**Personal Experience #3.** During a 3 a.m. maintenance session years ago, I thought I ran the command to replace a BGP policy with a new version. Instead, I replaced the *entire router* config with *only* the BGP policy, and then immediately saved this new 10-line router config. I earned double idiot points by doing it to both my primary and back-up router at the same time. Then for triple-word-score-brush-up-that-résumé bonus points I erred by doing the update in-band, not from the management console. Needless to say, I turned two beautiful Juniper peering routers into two of the nicest looking boat anchors one has ever seen.

However, not to despair. With JUNOS' 'rollback' command, I was able to slap a console cable in, and with **rollback 1** I was back online within two gut-wrenching minutes. With a non-Juniper router, I would not have been able to recover by simply rebooting to undo my changes, since I had over-written the operational config, and the routers themselves were my only way to get back to my offline router configurations repository. This example illustrates that if an engineer chooses to bypass all safety checks and commit an errant configuration, recovery with Juniper is available, and as simple as two commands: **rollback 1 / commit**. Additionally, had I used *commit confirmed*, the routers would have automatically saved operational trouble without any physical intervention on my part.

## Annotation and comments

JUNOS allows every line of the configuration to be annotated to explain to others on the team what each part of the configuration achieves. Additionally, JUNOS provides the ability to comment on the changes made during each commit, including who made the changes. Using **"commit comment"** engineers can tell other staff or members on other shifts what they trying to achieve with each configuration change. This allows for built-in change control and auditing, more accurate and timely recording, and reduction of troubleshooting delays and confusion.

When firewall filters are applied on other vendor platforms, after the first filter line is applied an implicit deny all is added at the end. This has caused many stressful evenings for engineers who could not enter firewall rules allowing their access into the running config fast enough before the router locked them out. With JUNOS' atomic commits, this problem disappears because all changes are made at once after you have the time to review them and confirm you're happy with the new configuration.

Another method by which JUNOS prevents pilot error is through simpler syntax for firewall filters, or Access Control Lists (ACLs) in common parlance. Syntax for ACLs for non-Juniper vendors is difficult to understand at best. With Juniper's ability to include detailed names for each stanza, apply names for subnets, and labels for custom ports, troubleshooting filter terms (or ACLs) becomes much easier and far less error prone.

For instance, here is a comparison of an ACL written for a firewall from Cisco, Foundry, or Force10—vs. the same filter syntax written in Juniper's JUNOS:

<b>Juniper Networks (JUNOS):</b>	<b>Cisco, others:</b>
<pre>firewall {   filter isp-inbound {     term discard-rfc1918-traffic {       from {         source-address {           10.0.0.0/8;           172.16.0.0/12;           192.168.0.0/16;         }       }       then {         count discarded-rfc1918;         discard;       }     }     term discard-bogons {       from {         source-address {           0.0.0.0/8;           127.0.0.0/8;           169.254.0.0/16;           224.0.0.0/3;         }       }       then {         count discarded-bogus;         discard;       }     }     term accept-log-and-count-all-else {       then {         count accepted-default;         log;         accept;       }     }   } }</pre>	<pre>ip access-list extended ISP-INBOUND deny ip 10.0.0.0 0.255.255.255 any deny ip 172.16.0.0 0.15.255.255 any deny ip 192.168.0.0 0.0.255.255 any deny ip 0.0.0.0 0.255.255.255 any deny ip 127.0.0.0 0.255.255.255 any deny ip 169.254.0.0 0.0.255.255 any deny ip 224.0.0.0 31.255.255.255 any permit ip any any log</pre>

**Within JUNOS, full routing tables are configured that leverage all of the same features and richness as the global routing table, allowing multiple ISPs to be blended together in any combination, thereby creating “virtual policy-based routers” within a single chassis.**

JUNOS also makes filter management less error-prone by eliminating the time-honored right-of-passage of destroying an entire list of firewall filters to change or re-order a single line. In other vendor platforms, a list of firewall filters must be deleted and reapplied from the beginning because entries can only append to the end. With firewalls that are thousands of lines long, this is a tedious task at best. When compared to the race condition minefield of “how do I keep the interface protected while I delete the ACL protecting it, and re-apply the new ACL all at the same time”?, the ability for engineers to modify individual lines, move sections of their firewall, re-order pieces, or insert new lines at any point, is a huge advantage.

## **Leveraging routing policies to achieve economic and business goals**

Many network designers look at BGP as a way to failover from one possible path to a destination to another candidate path to that destination in the event of a failure upstream. Early on in my career, I realized this is only the beginning of what BGP can accomplish and that BGP provides access to the so-called Layer “8” of the OSI model, the “economic” layer.

For “Layer 8,” BGP offers the ability to effect massive cost savings by intelligently leveraging inexpensive transport providers whenever possible, and more expensive operators only as necessary, to meet application performance requirements, thereby driving down the overall blended bandwidth rate.

Imagine your site has a small volume or percentage of total traffic that generates the most revenue for you (for example, ads and HTML) but a much larger volume of traffic that generates almost no revenue (for example, streaming media without embedded advertising). JUNOS provides the ability to create routing tables beyond the default global routing table called a Routing Information Base, or “RIB-Group”. With RIB-Groups and filters to identify different applications, latency-sensitive and highest priority traffic use the best performing (but most expensive) transport providers or ISPs, while all less important traffic uses more economical operators providing reduced performance, but still suitable to those applications.

This is not exactly the same as QOS (Quality of Service). QOS prioritizes the order in which packets leave the same interface in an effort to manage link congestion and application performance. RIB-Groups provide different next-hop addresses depending on traffic type and application needs. If QOS allows you to skip to the front of a line, RIB-Groups allow you to take the commuter lane vs. a rut-filled, rural dirt road.

## **Policy routing vs. RIB-groups**

Policy routing is when firewalls (filters) are applied to match on certain IP header characteristics and route only those matching packets differently from the default method. In non-Juniper routing environments, policy-based routing (PBR, also called FBE, Filter Based Forwarding) is typically achieved with static routes, without support for a full routing table. In such cases if the statically routed path is determined to be unavailable, all policy routed traffic is black-holed. The failover techniques available are typically limited to detecting link status, which is often “up” because of lit Metro Ethernet switches aggregating ports in front of the router moving traffic. In these situations, your failover capabilities are as robust as they are with static routing – not good.

RIB-Groups provide for a more robust policy routing environment by ensuring availability for the full set of network routes with fail-over routing for RIB-Groups and associated policies. Within JUNOS, full routing tables are configured that leverage all of the same features and richness as the global routing table, allowing multiple ISPs to be blended together in any combination, thereby creating “virtual BGP policy-based routers” within a single chassis.

**Personal Experience #4.** I have many times implemented RIB-Groups that directed all high-volume, but latency-insensitive protocols such as mail, FTP, and patch-updates out very inexpensive Tier-3 operators, while sending all ads and HTML out the fastest Tier-1 providers. The cost delta between these ISPs was over \$20/Mbps. When multiplied by many Gbps of traffic, the costs savings were many \$million per year. RIB-Groups allowed routing based on any firewall filter attribute such as source-IP or source-port, but most importantly provided something that no other vendor's policy based routing could: failover preserving the policy-based routing functionality in the dynamically updated topology.

Since other vendors set a default next-hop, they did not have an entire routing table as with RIB-Groups, so in both the high-performing and low-performing RIB-Groups I was able to use BGP to blend together multiple ISPs in each situation. I thought I could achieve the same policy with another vendor but I found out the hard way that policy routing does not re-route if it doesn't detect link failure (turns out the switch our interface terminated on was fine but the upstream BGP router was toast). In this non-Juniper implementation, the loss of the BGP peer was never detected and traffic continued to be black-holed until an engineer manually changed the policy. JUNOS in an identical configuration detected the loss of the BGP neighbor and rerouted out the remaining BGP peers with the same economic benefits.

## Better information and insight into your business

Imagine you run a website with fifty different properties managed by different groups on different hardware. If you're tasked with monitoring the network and you see a massive dip in bandwidth consumption, can you tell where the outage or reduction came from? There could be fifty sets of servers, load balancers, storage arrays, or other systems to check. Since the problem is with the distribution of applications over so many network components, it is impossible to know which system crashed, and a lot of time is wasted just trying to figure out where to start troubleshooting.

Juniper provides the highest level of network utilization detail and application usage with SNMP pollable firewall-based counters. JUNOS also allows the same counter name to be applied to multiple interfaces, aggregating the data with whatever method the customer needs.

**Personal Experience #5.** A CSP client had dozens of different websites that were each a bit fragile in their own way, and every day at least one of them would fail. The problem was the OS and application monitoring were poorly developed to triangulate on which property was having problems. By setting up an outbound count-only filter on each of the ISP uplinks, we were able to match on the source IP address for the load balancer VIPs in each property to know how much traffic each site was producing. If a single site had multiple VIPs, it was trivial to include all of them within the single counter for each property. We could apply this identical outbound filter to every ISP uplink and we received a perfect breakdown of the total outbound traffic. Anytime there was a dip in traffic, we knew immediately which of the VIPs and associated application/server farms experienced an outage regardless of which ISP the traffic was egressing to.

**Personal Experience #6.** Another CSP client had a very large network that provided transit services for other sister companies. Their challenge was to determine who should pay for what percentage of the transit bill each month. When a sister company plugged into the larger backbone, some of the traffic went out via an ISP and some of it stayed on the backbone, going to another internal datacenter, and no one knew how to allocate costs appropriately. By applying an outbound filter with company-specific counters on each ISP that matched on the source-address subnet, we were able to total how much bandwidth each company sent to each ISP, and bill back accordingly. We also subtracted the total transit usage from the backbone egress and calculated a different bandwidth charge for traffic across the much cheaper backbone circuits to recoup costs that were otherwise

absorbed by the business unit running the backbone. In the end, the company was able to bill every user of the network and no company unit assumed an unfair economic burden while another received a free ride.

**The same code that runs on the smallest J-series and M7i is built from the same source code train that builds releases for the largest T1600 or MX960.**

## Platform stability: The JUNOS Competitive Advantage

Many networks use bleeding edge code out of necessity for a particular bug fix, feature, or possibly out of a pathological desire to test Murphy's Law. Everything described in this paper I have personally used on Juniper routers since 2001. All of this code is well tested, well supported and well documented, and proven to work in today's large networks.

The last (but far from final) method we'll discuss focuses on how JUNOS decreases the likelihood of pilot error is by providing a single code train for the entire Juniper product family (M, T, MX, EX, and J series platforms). The same code that runs on the smallest J-series and M7i is built from the same source code train that builds releases for the largest T1600 or MX960. This single, multi-platform code train, combined with the modularity intrinsic to the FreeBSD OS, means Juniper can deliver more feature updates, with easier maintenance, and less error than any other routing vendor platform. The Juniper code I run on my customers' MX960, MX480, M320, M20, M10i--and even my personal, seven-year-old M5 router--is all the same and built upon the same well-proven and time-tested JUNOS software.

JUNOS offers a rich set of features and options not available in any other routing platform. When combined with creative network engineering resources, JUNOS allows companies to tap into a wealth of cost savings, redundancy improvements, simplification of routing topology, and removes many causes of pilot error that are simply not possible with other vendors.

The Juniper platform is far and away the best performing routing hardware that allows me and my engineering team at Peak Web Consulting unparalleled control to go beyond relatively simple request of "keep the network running" to leveraging strategic advantages for our customers over their competition.

With Juniper Networks routers, Peak Engineers are able to implement more changes faster and with less error than with any other routing platform. This directly translates to providing our Juniper customers with the ability to scale faster and take full, dynamic advantage of the infrastructure hardware they introduce.

Juniper Networks customers are also smarter about the operation of their networks. They know where their traffic is going, what path it takes to get there, and most importantly, what other options exist. Juniper customers are able to make better ISP selection decisions with flow analysis that does not cripple their routing performance, and do so with improved visibility into what traffic is filling their individual pipes via SNMP polling methods unavailable on any other platform. With this improved intelligence, ISPs and peers are no longer selected based on their price, but rather on how they service the targeted eyeballs Peak's customers are trying to reach.

The single largest advantage for Peak's Juniper customers is their decreased cost of operations for delivering their product. The ability to tailor the cost and performance of an ISP to individual applications provides a unique competitive advantage for any company.

## What are your challenges?

I hope you found this paper useful in planning or enhancing your content delivery network.

If you have related questions on this content or issues you would like to discuss, you can contact me at [Jeffrey@peakwebconsulting.com](mailto:Jeffrey@peakwebconsulting.com).

Jeffrey Papen, Founder, Peak Web Consulting. JNCIE #116,

*Certified Juniper Networks Instructor*

Jeffrey Papen implemented the BGP load-balancing and multihoming policies at both Yahoo! and Excite@Home from 1998 to 2003. His responsibilities at Yahoo! included developing the BGP multihoming policy to optimize network performance and balance traffic levels to over 200 peering and transit ISPs, while meeting transit commitments within 1 percent of their theoretical minimums, and still maintaining the highest possible performance.

Mr. Papen has also developed numerous network analysis tools including SQL-based bandwidth usage and billing reconciliation (happydog), ISP SLA testing (Glacier), BGP transit analysis (TUNDRA), and non-invasive multihoming performance testing (Alpine). Since leaving Yahoo! in March of 2003, he has managed Peak Web Consulting, whose team of engineers provide networking services to some of the largest Web 2.0 companies.

---

