Juniper Networks | Engineering Simplicity

Juniper Scale-Out Stateful Firewall and CGNAT for SP Edge — JVD

Published
2025-05-29

Table of Contents

Juniper Scale-Out Stateful Firewall and CGNAT for SP Edge — JVD

Juniper Networks Validated Designs provide you with a comprehensive, end-to-end blueprint for deploying Juniper solutions in your network. These designs are created by Juniper's expert engineers and tested to ensure they meet your requirements. Using a validated design, you can reduce the risk of costly mistakes, save time and money, and ensure that your network is optimized for maximum performance.

About this Document

This document explains a Juniper Validated Design (JVD) for the Scale-Out Security Services solution, which can be deployed at the SP multiservice edge WAN or metro networks. It validates the network services complex consisting of MX universal services routers coupled with Juniper SRX/vSRX Series Firewalls delivering carrier grade NAT (CGNAT) and stateful firewall function in verity of deployment scenarios.

The summary of the solution platforms is as follows:

Table 1: Solution Platforms Summary

Solution	Forwarding Layer	Service Layer
Scale-Out Security Services for SP MSE	MX304 Universal Edge Router	SRX4600 vSRX

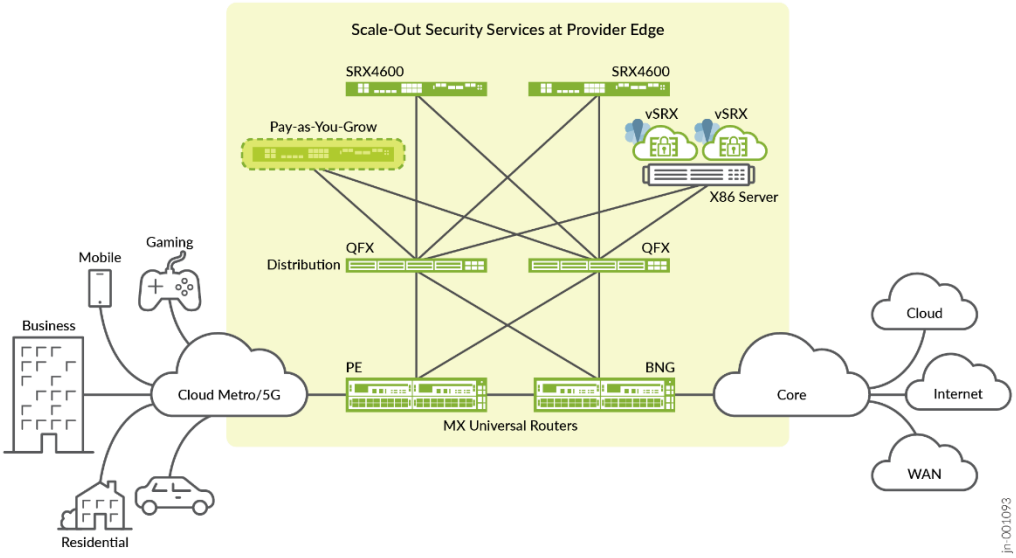
Solution Benefits

The Juniper Scale-Out Security Services solution is a common security services complex featuring a Stateful Firewall (SFW) and Carrier Grade Network Address Translation (CGNAT) for use in a fixed and wireless Multiservice Edge (MSE) and Broadband Edge (BBE) deployments for Service Providers and MSO’s. The security complex leverages the scale-out network architecture and automation with a tight integration between routing and security services elements represented by MX universal routers and

SRX Series Firewalls. This provides the best routing and security stacks for optimal performance and total cost of ownership. The scale-out approach offers advantages over scale-up and integrates security engines directly into the routing nodes, including:

- Highly scalable CGNAT/SFW systems with respect to number of traffic flows and IPv4/IPv6 prefixes
- Pay-as-you-grow approach
- Flexibility to handle unpredictable traffic growth
- High availability with sub-second restoration for stateful traffic flows
- Optimal operational preferences for a choice of physical or virtual nodes
- Improved time to market security services on new platforms
- Flexible placement of security services in the network

Figure 1: Juniper Scale-Out General Architecture



This solution is equally applicable for the green-field deployments or as a nested solution on top of the existing MX-series routers in the centralized or distributed multiservice edge segment of SP networks allowing flexibility in placement of the services across SP WAN infrastructure.

The Scale-Out Security Services solution provides a scale-out model for enabling high capacity CGNAT and SFW services combining Juniper MX Series modular and compact routers with Juniper vSRX and SRX4600 security products (Virtual Network Functions or Appliances). In general, a solution includes three layers: forwarding layer, security services layer, and management and control layer. These layers

enable consistent traffic flows through the service complex in both directions, addressing high availability requirements and simplified operations and management of multiple systems constitute the solution.

This JVD focuses on the first two layers only, which include the following functional elements and solution building blocks:

- Security Services Layer:
  - CGNAT
  - Stateful Firewall
  - High availability function
- Forwarding Layer:
  - PE forwarding plane with virtual routing instance (“external” and “internal”)
  - Load balancing between multiple nodes of the service layer
  - High availability function
  - May include a distribution-forwarding layer optionally

Use Case and Reference Architecture

IN THIS SECTION

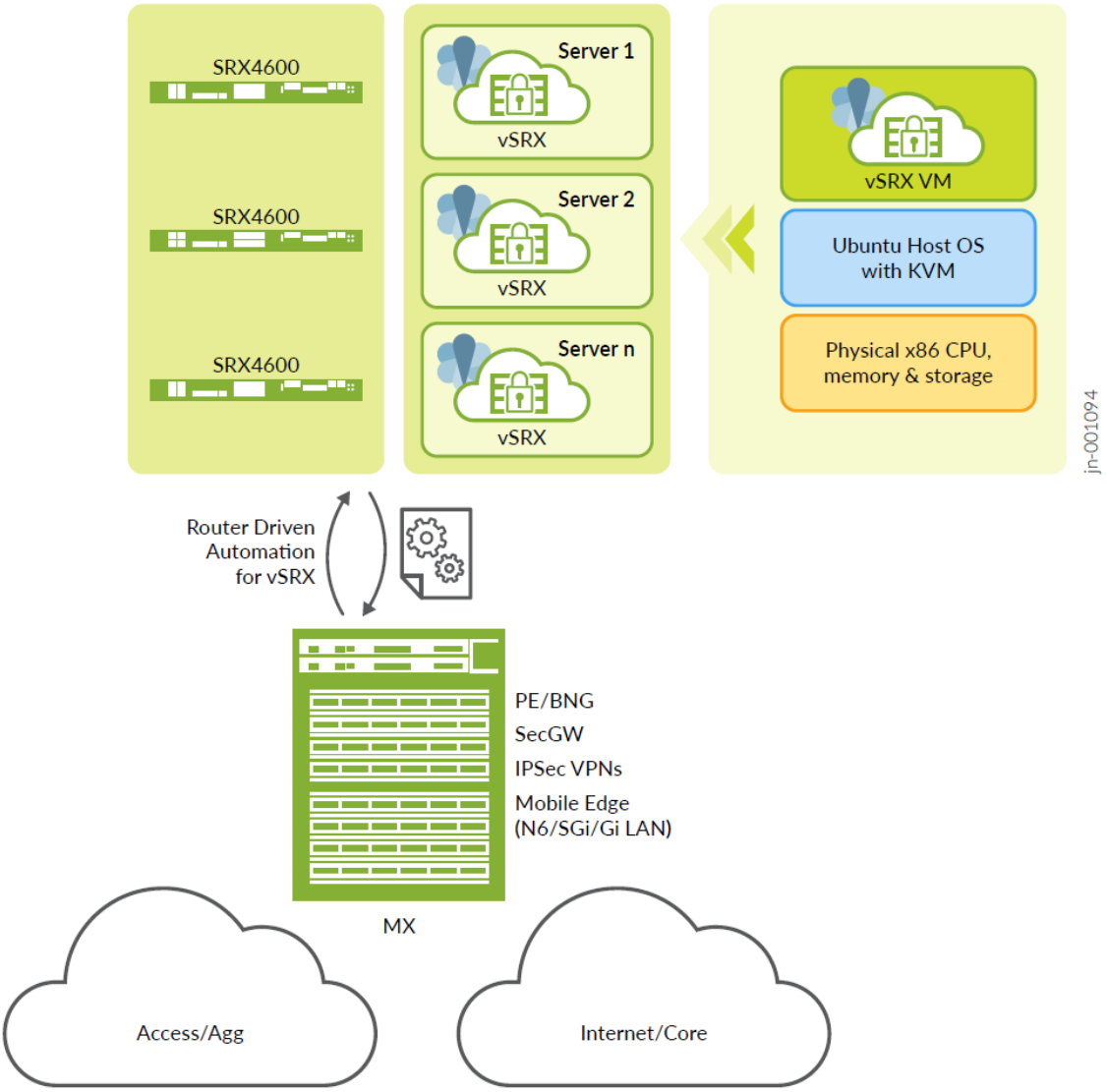
Solution Functional Elements

Juniper Off-Box Security Services solution architecture includes two main functional blocks:

The security services device formed by a standalone vSRX virtual network functions or SRX4600 or a redundant pair of the same device. This section focuses on the standalone use case, former section shares details on the redundant solution architectures.

The MX Series Router as load balancer router - The Juniper MX Series Routers provide 100G or 400G interfaces to the servers hosting vSRXs or the SRX4600s forming the complex of services. Both access side and Internet side peering (see Figure 2 on page 5 for reference) are enabled through MX Series Router dedicated ports being used for high throughput.

Figure 2: Scale-Out Solution Functional Blocks



With the new Trio 6 MX10004 and 10008 systems, capacity per slot is up to 9.6 Tbps and with compact MX304 systems, capacity per slot is up to 4.8 Tbps, enabling a high number of 100G ports. An MX304 router can provide up to 48 x 100G interfaces and an LC9600 line card in a modular MX10000 system, up to 96 x 100G ports.

To optimize port usage, it is recommended to implement an intermediate distribution layer with two (or more) QFX-series switches to aggregate multiple SRX/vSRX Series Firewalls nodes into a bundled 400GE links on the MX Series Router.

If vSRX firewall is the choice for the security element, it can be rolled out on top of the KVM or VMware virtual network function, running on open compute servers. You can bring your own server based on

prescribed server specifications (CPU cores, memory, Linux OS, KVM versions). For more information about the server specifications, see **vSRX server specifications** in the references.

vSRX is a Virtual Network Function (VNF) running on KVM or VMware hypervisors, with a flexible compute server allocated by number of cores (up to 32) and memory (up to 64G). Networking wise vSRX can use virtio or SR-IOV with smart NICs like Mellanox ConnectX-6.

A complete Off-Box solution requires implementation at three fundamental layers: Data, Control, and Management layers. This solution enables consistent traffic flow through the service complex in both directions, addresses high availability requirements, and simplified operations and management of multiple systems.

For this JVD, an external BGP (eBGP) protocol with BFD provides a routing and control function between network elements of the complex while implementing load balancing with two approaches:

- Equal Cost Multi-Path (ECMP) load balancing function with Consistent Hashing (CHASH)
- RE based traffic load balancer function (TLB) on MX Series Router

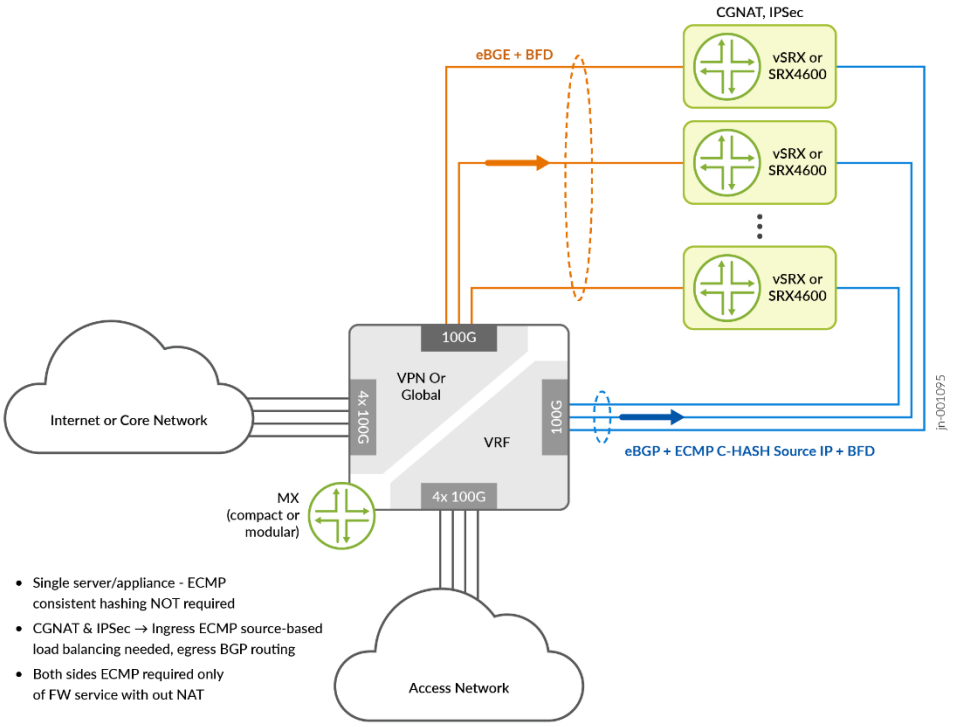
Two routing instances – Access and Internet – are used on MX Series Router to peer with corresponding network segments of the SP network infrastructure and the security node. eBGP enables scalable and flexible exchange of routing information for the access and the Internet side routing (see Figure 3 on page 7). The failure detection is based on BFD with timers as low as 100ms, enabling fast reconvergence and fast and automatic adjustment for the ECMP load balancing.

To maintain higher level of security in future applications like Managed Enterprise Firewall service - where injection of your routes into the security layer is not preferred - static routes with BFD protection are the preferred control and traffic distribution method.

The access side traffic is load balanced between services nodes dynamically based on ECMP with source IPv4 or IPv6 addresses consistent hashing. For the CGNAT and SFW on the Internet side, eBGP routing and BFD failure detection is required. Destination based IPv4 or IPv6 ECMP consistent hashing (CHASH) is used on the Internet side with stateful firewall services without NAT.

Essentially ECMP with CHASH limits the impact on existing traffic flows in the event of service node failure or addition of new service node to the complex. On service node failure, impacted events flows are rehashed and rebalanced, while on addition of new service modes, limited equal number of flows from each member in cluster are rehashed and rebalanced in the new member in the cluster, limiting the impact while maintaining the equal cost load balancing.

Figure 3: CHASH Based Network Architecture



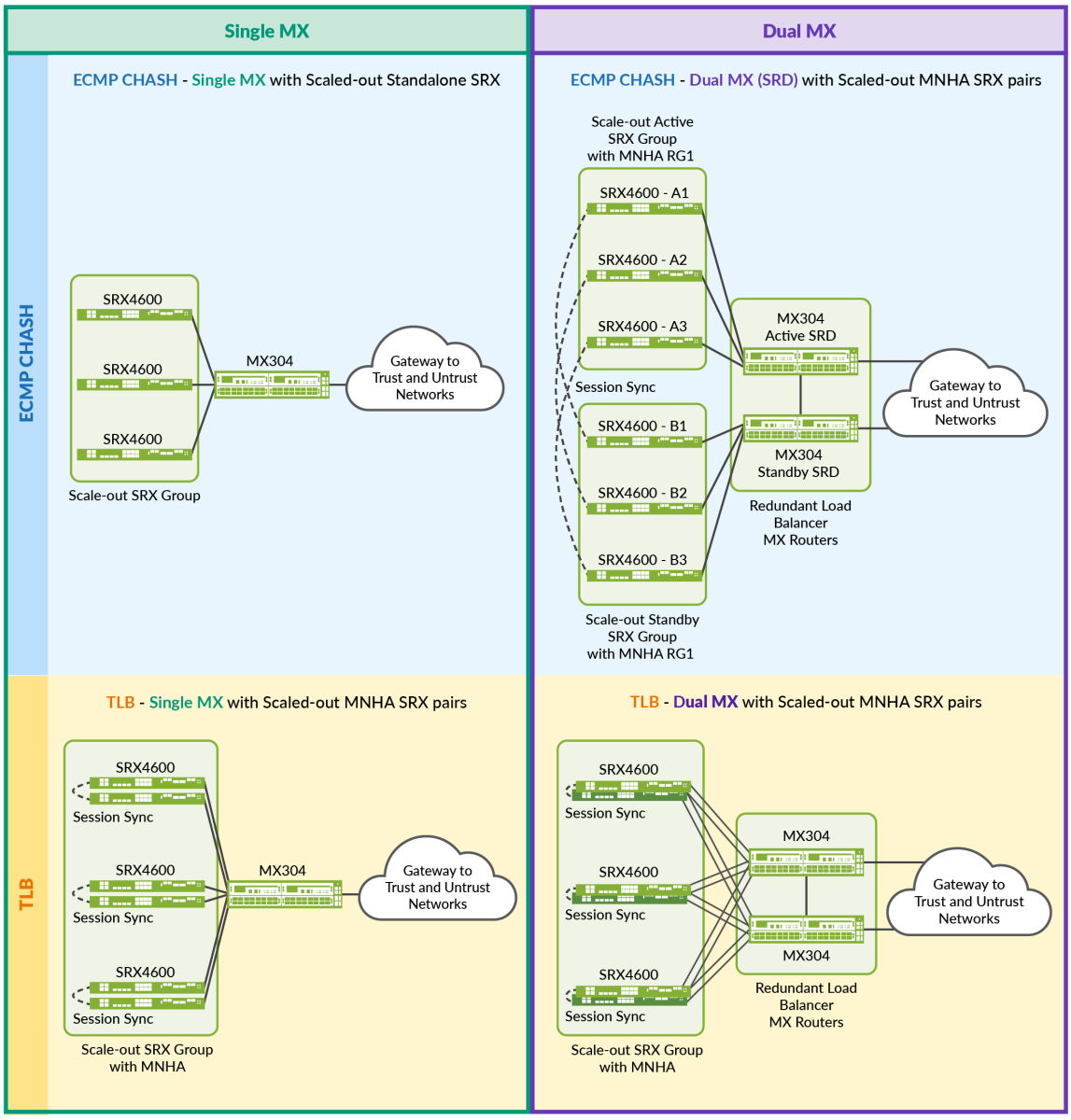
This architecture allows you to scale the service complex with tens of service nodes (SRX Series Firewalls /vSRX) with efficient load balancing of flows between service nodes and minimizing the effect (blast radius) due to a single node failure. The eBGP routing on MX Series Router in its turn scales beyond Internet tables to millions of routes if required and easily beyond.

Solution Deployment Scenarios

Following the suggested solution architecture, a few deployment scenarios are considered where MX Series Router and SRX Series Firewalls are connected in either standalone or redundant pairs (see topologies). The architecture uses network redundancy mechanisms to provide flow resiliency between the MX forwarding layer and SRX Series Firewalls services layer (MNHA, aka L3 cluster, is explained later in the document). On dual MX with ECMP, a Service Redundancy Daemon (SRD) is used to monitor failure events to trigger a failover to the second MX Series Router. Note this is not required with TLB. Also, BFD protocol is used to achieve a quicker failover mechanism on routing when any other failure occurs. If SRX Series Firewall MNHA provides session synchronization (stateful sessions) between two nodes, then existing traffic and tunnels can continue to operate uninterrupted.

The following diagram shows the four main topologies covered in this JVD, combining standalone/dual MX with standalone/MNHA for SRX Series Firewall, each on a particular load balancing mechanism (ECMP or TLB). It uses three SRX Series Firewalls for the first topology and doubles them to three pairs for the other topologies.

Figure 4: Validated Topologies



There are numerous trade-offs with each of the architectural choices. In general, complexity increases as more redundancy is added. For example, SRX Series Firewall MNHA pairs introduce some requirements like a network link for HA communications. There are also dependencies on which load balancing

method is used on the MX Series Router (namely ECMP CHASH or TLB). This selection of topologies covers the most important considerations of simple to more redundancy scenarios.

- ECMP CHASH is simple to use, leverages standard protocols and well known ECMP mechanism, which might be a preferable option for some SP or enterprise network operations department, though this method is limited when it comes to failover capabilities.
- TLB has load balancing capabilities (at the time of publishing this JVD), which leverages services to load balancing, offers better redundancy capabilities, and can be multiplied with different local groups. It is useful when you need to combine different use cases on the same architecture. This method may not be backward compatible with older Junos releases.

Table 2: Validated Features Combination

Load Balancing Method	Junos for MX	Number of MX Routers	Security Features	SRX in MNHA Cluster Mode	SRX in Standalone Mode
ECMP with CHASH	23.4R2	Single MX	SFW/CGNAT	No	Yes
		Dual MX (SRD)	SFW/CGNAT	Yes	No
Traffic -Load - Balancer [TLB]	23.4R2	Single MX	SFW/CGNAT	Yes	Yes
		Dual MX with Health Checking	SFW/CGNAT	Yes	Yes

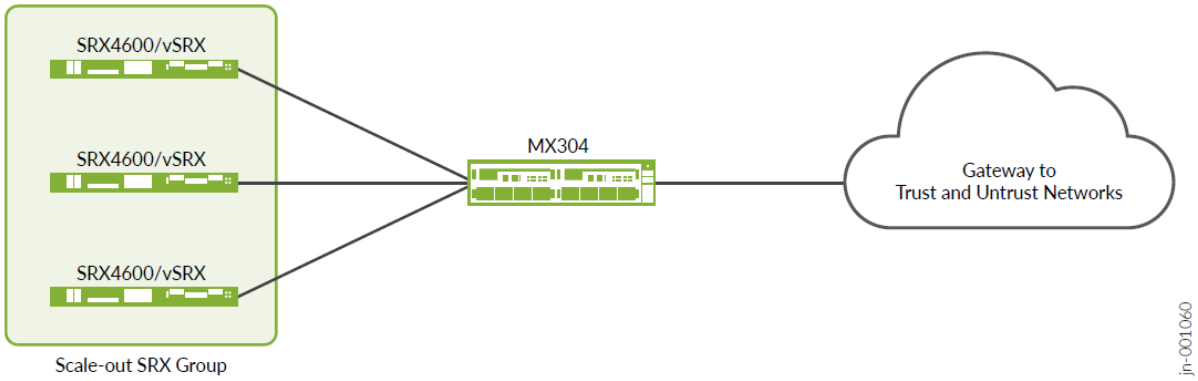
Note that the scale-out solution only uses standard mechanisms and protocols between the components and does not require any special proprietary protocols. The exception is how load balancing is implemented internally (how the MX Series Router handles and distributes sessions). From a networking point of view, this solution uses standard protocols.

Following are some recommendations that may help you in selecting the deployment method.

Deployment Scenario 1 – ECMP CHASH – Single MX Router with Scaled Out Standalone SRXs (Multiple Individual SRX Series Firewalls)

This topology is simple and least redundant. The resiliency is provided at MX Series Router, with a redundant RE, PSU, etc however, there is no protection against MX-node failure. Deployment provides protection against service node failure by redistributing traffic flows between two remaining security nodes. Though there is no session synchronization between the SRX Series Firewalls, which leads to longer restoration time for the affected flows.

Figure 5: Deployment scenario 1 – ECMP CHASH - Single MX, Standalone SRXs



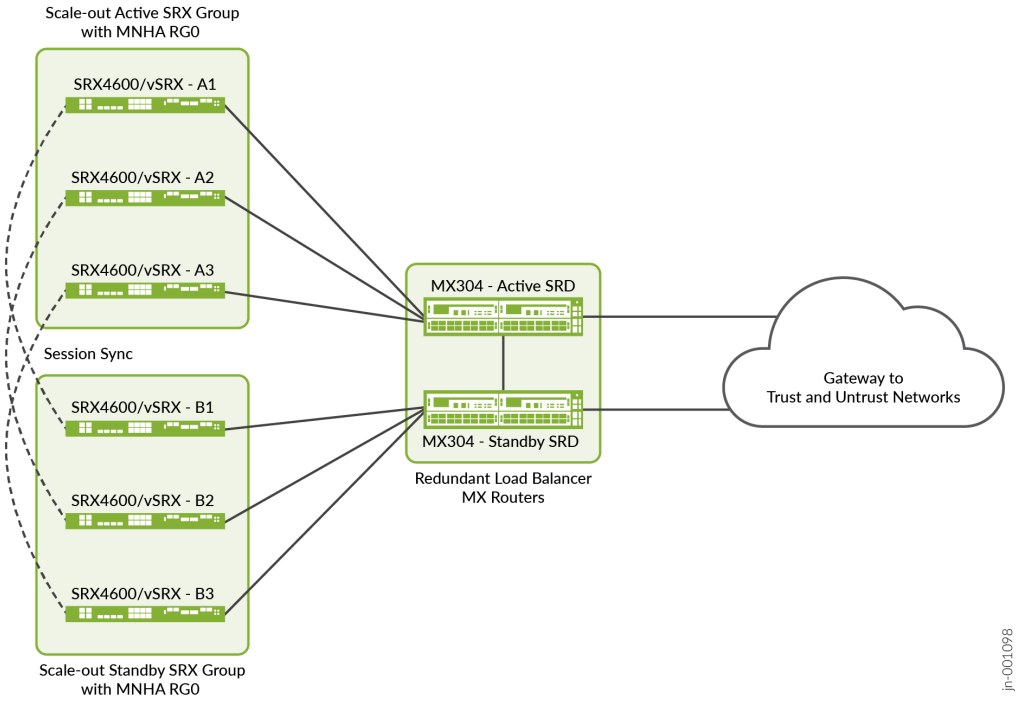
Network operators that are not concerned about stateful failover may want to simply augment security service capacities by adding more SRX Series Firewalls. The application sessions may be short lived anyway (for example, a redundancy mechanism may be handled at the application level so session sync between two different firewalls is not required).

- Pros: Simplicity and scaling with each individual SRX Series Firewalls
- Cons: No redundancy

Deployment Scenario 2 – ECMP CHASH – Dual MX with Scaled Out MNHA SRX pairs (Multiple Pairs of SRX Series Firewall)

This topology does offer redundancy at both the MX Series Router and for each redundant SRX Series Firewall pair. The redundant pair of MX Series Router uses an SRD mechanism providing monitoring of physical elements of the network and/or the MX Series Router itself, as well as any other routing and system events that may need to trigger a failover to the other MX Series Router.

Figure 6: Topology 2 – ECMP CHASH - Dual MX with SRD, SRX MNHA Pairs



In case of a network failure detected by the active MX Series Router, the second MX Series Router takes over the active role and all traffic is then redirected to this active MX Series Router. It means that traffic is sent to the previous backup SRX Series Firewall, becoming master of the MNHA pair. This architecture only allows use of one SRX Series Firewall of a pair at a time, basically the SRX Series Firewalls connected to the same MX Series Router. However, in case of any failover, the traffic continues across the second node of each MNHA pair.

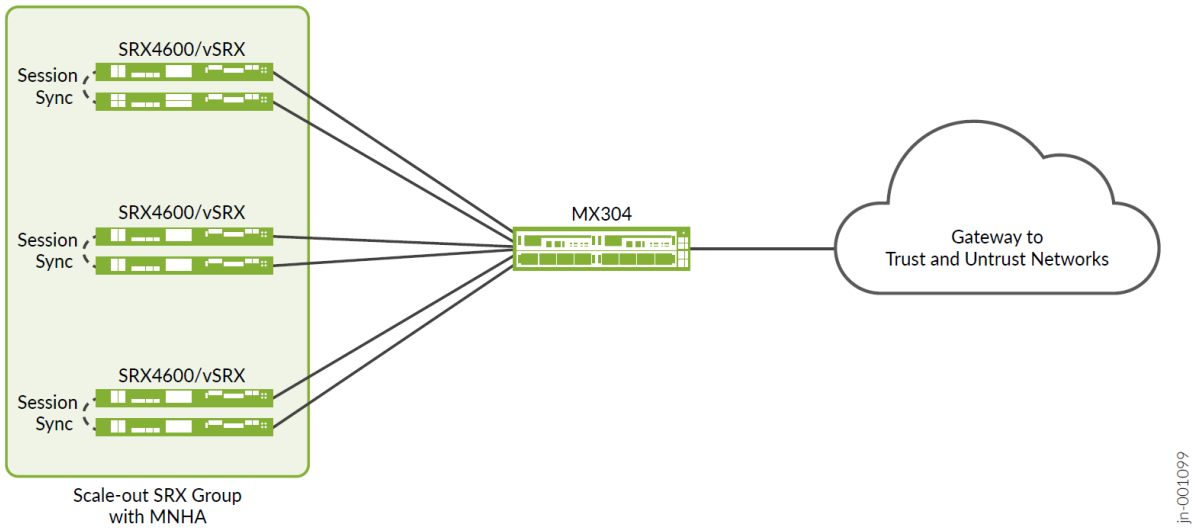
On the SRX Series Firewalls side, MNHA allows both SRX Series Firewalls to handle and synchronize sessions and support any requested security services on both firewalls. Since this topology uses SRG0 as cluster mode, there is no need of failing over a SRX Series Firewalls to the other firewall in case of any failure detected by the MX Series Router (only when detected by SRX Series Firewalls itself). The session synchronization allows any traffic coming from the MX router (at SRD level) to process traffic for existing sessions, and any new sessions coming to it.

- Pros: Simple redundancy and scaling with each SRX Series Firewalls pair
- Cons: half of the architecture is active at a time

Deployment Scenario 3 – TLB – Single MX Scaled Out MNHA SRX Pairs (Multiple Pairs of SRX Series Firewalls)

This topology does offer redundancy for the SRX Series Firewalls however, not for the MX Series Router, though this one may have a second Routing Engine (RE) installed in the appropriate slot and is not using two MX chassis in that case.

Figure 7: Topology 3 – TLB - Single MX, SRX MNHA Pairs



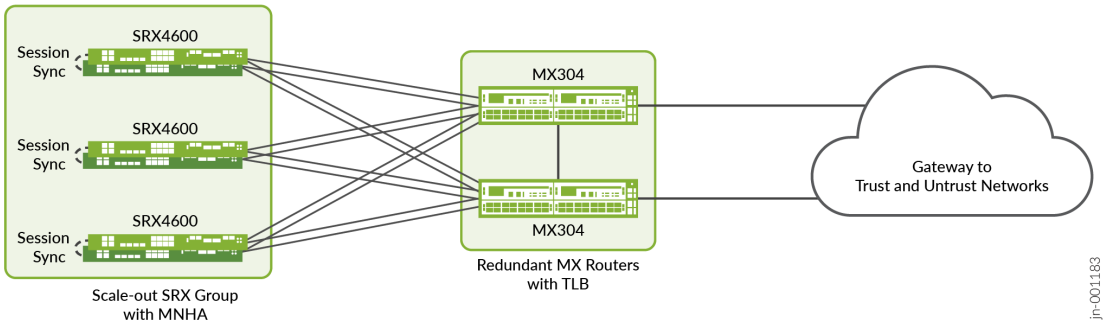
MNHA offers sessions synchronization within a cluster and help with any failure scenario.

- Pros: Redundancy and scaling with each SRX Series Firewalls pair
- Cons: No redundancy on the router (except using dual RE)

Deployment Scenario 4 – TLB – Dual MX Scaled Out MNHA SRX Pairs (Multiple Pairs of SRX Series Firewalls)

This last topology offers the most redundancy for both MX Series Router and SRX Series Firewalls nodes and takes advantage of having all components used at the same time. Any failover scenario can be covered.

Figure 8: Topology 4 – TLB - Dual MX, SRX MNHA Pairs



MX Series Routers handle traffic on any of the two routers, while SRX Series Firewalls can be used either in Active/Backup role or in Active/Active role, making use of both nodes at the same time. This augments the capacity of the network during normal operation, however this leaves one active role at a time when a failure occurs (consider a single MNHA cluster).

Each SRX Series Firewall is connected to both MX Series Routers. If any of one node fails within a cluster, all other SRX Series Firewalls pairs might have an independent failover from the other SRX Series Firewalls pairs and the MX Series Router.

- Pros: Full redundancy and scaling for MX Series Router and SRX Series Firewall pairs.
- Cons: More interfaces used on the MX Series Router if directly connected. Then, an optional distribution layer can cover more connectivity needs when SRX Series Firewall count augments.

Validation Framework

IN THIS SECTION

Test Bed Topology

In this JVD, two physical topologies are leveraged for standalone (Figure 9 on page 14) and redundant configurations (Figure 10 on page 15) are able to address all four deployment scenarios as described in "Supported Platforms" on page 15. As mentioned in the configuration example section, some key elements need to be put in place, like a consistent network IP address scheme, the BGP peering between the MX Series Router, the external Gateway (if any), and with each SRX/vSRX Series Firewalls.

Figure 9: Test Bed – ECMP CHASH - Single MX, Standalone SRXs

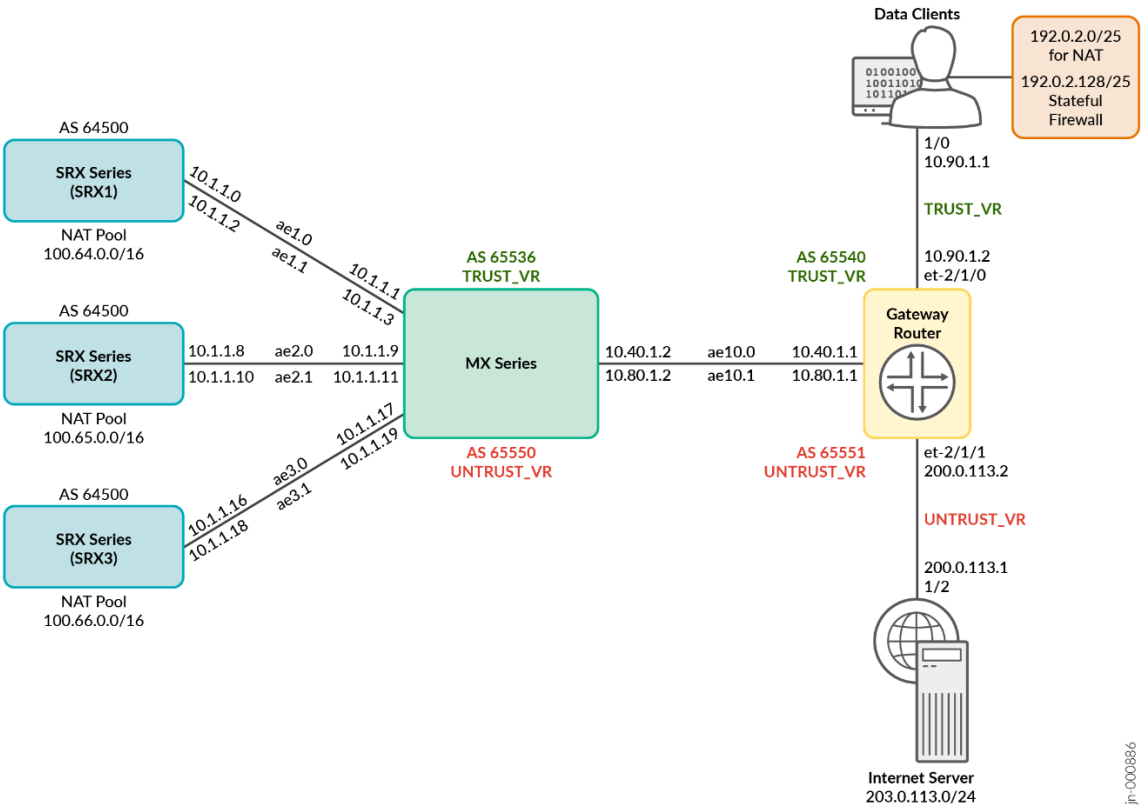
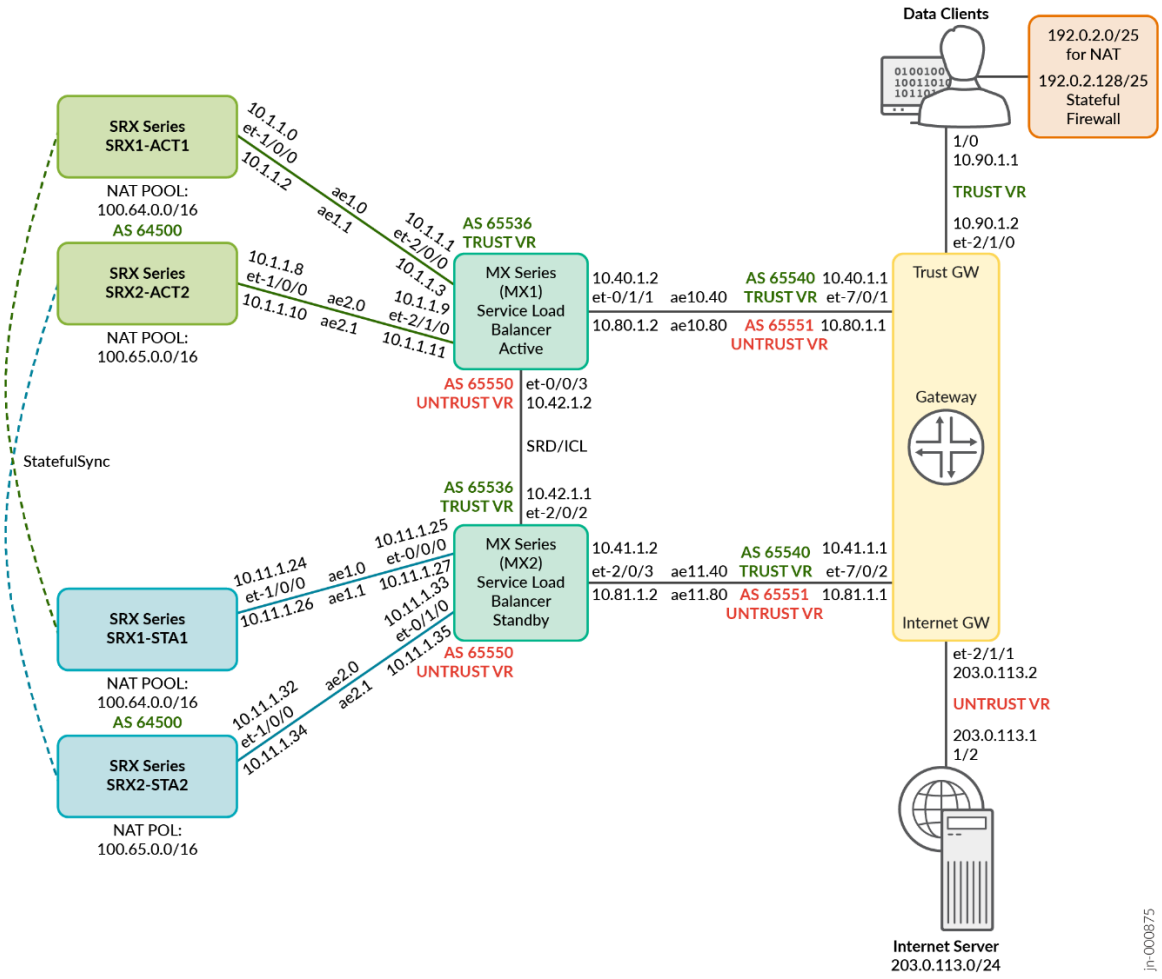


Figure 10: Test Bed – ECMP CHASH - Dual MX, SRX MNHA Pairs



Supported Platforms

To review the software versions and platforms on which this JVD was validated by Juniper Networks, see the Validated Platforms and Software section in this document.

Tested Optics

The Fiber optic transceivers used in that test bed are:

- QSFP-100GBASE-SR4: between MX304 and SRX4600s
- QSFP28-100G-AOC-3M: between MX304 and servers hosting vSRXs

This JVD has been validated with the fiber optics reference above, but the technical validation is larger regarding hardware compatible optics, see those refs on Juniper's Hardware Compatibility Tool.

- For SRX4600: https://apps.juniper.net/hct/product/?prd=SRX4600
- For MX304: https://apps.juniper.net/hct/product/?prd=MX304
- For MX10004: https://apps.juniper.net/hct/product/?prd=MX10004

vSRX Setup and Sizing

This JVD focuses only on the functional aspect of the solution. It does not matter whether powerful servers are tested for hosting the vSRX(s), as well as the size of vSRX used here. For real time performances, high end servers (like Dell or HPE servers with Intel Gold or AMK 9K CPUs, 256GB RAM and ConnectX6 or X7 or later interfaces) with large vSRX sizes are proposed (like 16 vCPU and 32GB RAM). For more information about vSRX requirements, see Juniper documentation:

https://www.juniper.net/documentation/us/en/software/vsrx/vsrx-consolidated-deployment-guide/vsrx-kvm/topics/concept/security-vsrx-kvm-understanding.html

or

https://www.juniper.net/documentation/us/en/software/vsrx/vsrx-consolidated-deployment-guide/vsrx-vmware/topics/concept/security-vsrx-vmware-overview.html

Test Objectives

IN THIS SECTION

JVD is a cross-functional collaboration between Juniper solution architects and test teams to develop coherent multidimensional solutions for domain-specific use cases. The JVD team comprises technical leaders in the industry with a wealth of experience supporting complex use cases. The scenarios selected for validation are based on industry standards to solve the critical business needs with practical network and solution designs.

The key goals of the JVD initiative include:

- Validate overall solution integrity and resilience
- Support configuration and design guidance
- Deliver practical, validated, and deployable solutions

A reference architecture is selected after consultation with Juniper Networks global theaters and a deep analysis of use cases. The design concepts that are deployed use best practices and leverage relevant technologies to deliver the solution scope. KPIs are identified as part of an extensive test plan that focuses on functionality, performance integrity, and service delivery.

Once the physical infrastructure required to support the validation is built, the design is sanity-checked and optimized. Our test teams conduct a series of rigorous validations to prove solution viability, capturing, and recording results. Throughout the validation process, our engineers engage with software developers to quickly address any issues found.

The test objective is to validate the scale-out architecture, showing the various topologies with single/dual MX Series Routers and multiple SRX Series Firewalls, and demonstrate its ability to respond to various use cases while being able to scale. The different possibilities offered by routing, and the two main load balancing methods, using different platform sizes for MX Series Router and/or SRX Series Firewalls, using high availability of the various components.

Additional goals demonstrate scale-out capability of the solution, which allows linear performance and logical scale (stateful traffic flows) growth in the process of new SRX/vSRX Series Firewalls addition to the security services complex.

This JVD validates system behavior under the following administrative events, with a general expectation to have no or little effect on the traffic:

- Adding a new SRX series firewall to the service layer helps in redistribution of traffic to get an even distribution, no traffic disruption expected for other traffic.
- Removing a SRX Series Firewalls from the service layer causes traffic redistribution only for those associated to this removed SRX Series Firewalls.
- Having a SRX Series Firewalls failover to its peer (MNHA case) and returns to a normal state cause no traffic disruption and preserves sessions and IPsec Security Associations.
- Having an MX Series Router failover (dual MX Series Router) causes no traffic disruption.
- Varying themes and failure scenarios cause no traffic disruption.

The following networking features are deployed and validated in this JVD:

- Dynamic routing using BGP
- Dynamic fault detection using BFD
- Load balancing of sessions across multiple SRX Series Firewalls in standalone or high availability
- Load balancing using ECMP CHASH, first appeared in Junos 13.3R3
- Load balancing using TLB on the MX Series Router (TLB, first appeared in Junos 16.1R6)
- MX Series Router redundancy using SRD between two MX Series Routers with ECMP CHASH
- MX Series Router redundancy using BGP dynamic routing between two MX Series Router with TLB
- SRX Series Firewalls redundancy using MNHA as Active/Backup with sessions synchronization
- Dual stack solution with IPv4 and IPv6
- SFW is validated with simple long protocol sessions (HTTP, UDP)
- CGNAT is using NAPT44

Test Non-Goals

Maximum scale and performance of the individual network elements constitute the solution. There is no preferred specification for the hypervisor hosting the vSRX firewall, nor any specific vSRX sizes (in vCPU/vRAM/vNIC quantity). Simple vSRX firewalls are enough for testing the features. Note that vSRX firewall runs on many hypervisors including: ESXi, KVM, and Microsoft for onprem. Though vSRX firewall can also be deployed in public clouds like AWS, Azure and GCP, the purpose of the architecture is not to run with vSRX firewall in those external clouds where it may be questionable to consider the networking plumbing to get them connected.

> **NOTE**: This JVD does not mention about automation. However, automation is used to build and test the solution with various use cases and tests.

Following features and functions are not included in this JVD:

- Automated onboarding of the vSRX firewall
- Security director
- Network Address Translation: NAT64, DetNAT, PBA, DS-Lite
- Load balancing: filter-based forwarding
- Application and Advanced Security features like AppID, IDP, URL filtering, and another Layer 7

Tested Failure Events

SRX Series Firewalls failure events:

- MX Series Router to SRX Series Firewall link failures
- SRX Series Firewall reboot
- SRX Series Firewall power off
- Complete MNHA pair power off

MX failure events:

- Reboot MX Series Router
- Restart routing process

- Restart TLB process in MX router (traffic-dird, sdk-process and netmon deamon)
- GRES (Graceful Restart of routing daemon)
- ECMP/TLB next-hop addition or deletion (adding or deleting a new scale-out SRX MNHA pair)
- SRD based CLI switchover between MX Series Router (ECMP)

Traffic recovery is validated post all failure scenarios.

UDP traffic generated using IxNetwork for all the failure related test cases is used to measure the failover convergence time.

Tested Traffic Profiles

Tested traffic profiles are composed of multiple simultaneous flows for either a standalone SRX Series Firewall or a SRX MNHA pair in Active/Backup mode.

Table 3: Tested Traffic Profiles per SRX Series Firewall Pair

CPS/MNHA-Pair	Throughput/MNHA-Pair	Traffic Type	File Size
N/A	100Gbps	TCP	4k
N/A	100Gbps	UDP	IMIX
100K	N/A	TCP	1byte

Packet size is using Internet mix with average packet size of ~700bytes. The Packet Size:Weight distribution is as follows:

- 64:8
- 127:36
- 255:11
- 511:4
- 1024:2
- 1518:39

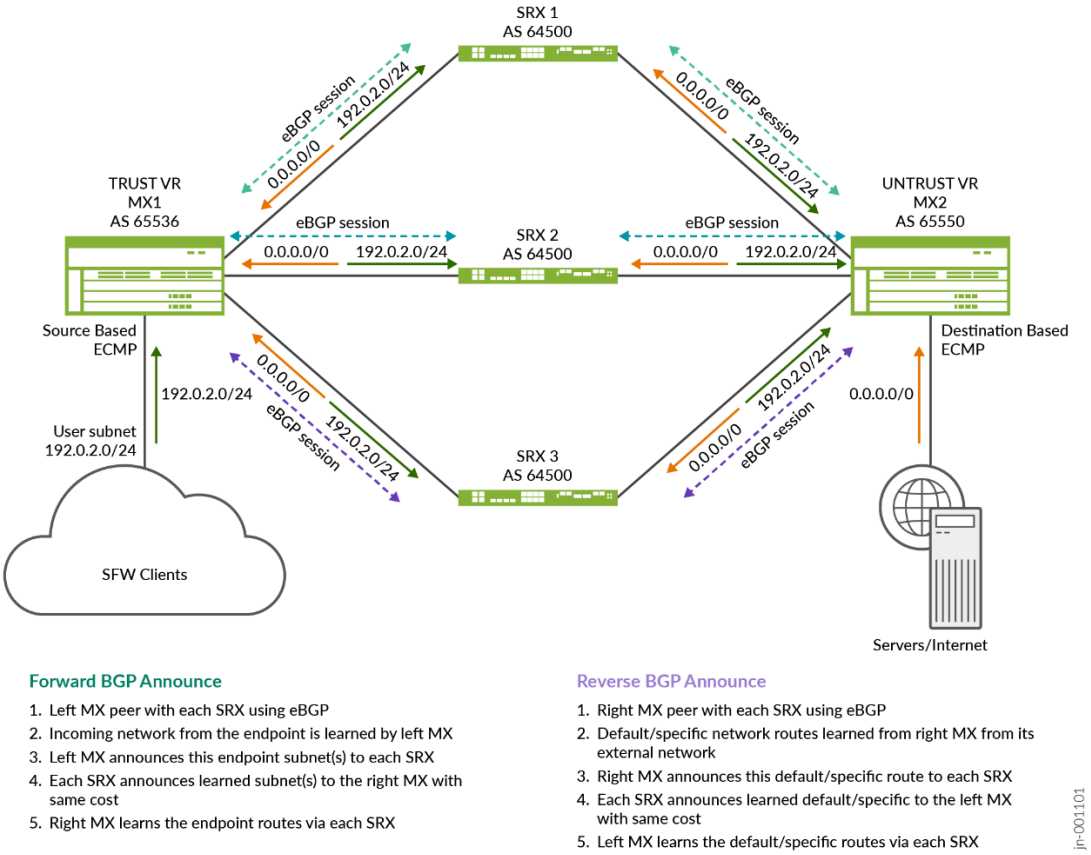
Test Bed Configuration

Contact your Juniper representative to obtain the full archive of the test bed configuration used for this JVD.

Traffic Path in SFW and CGNAT Scale-Out Solution

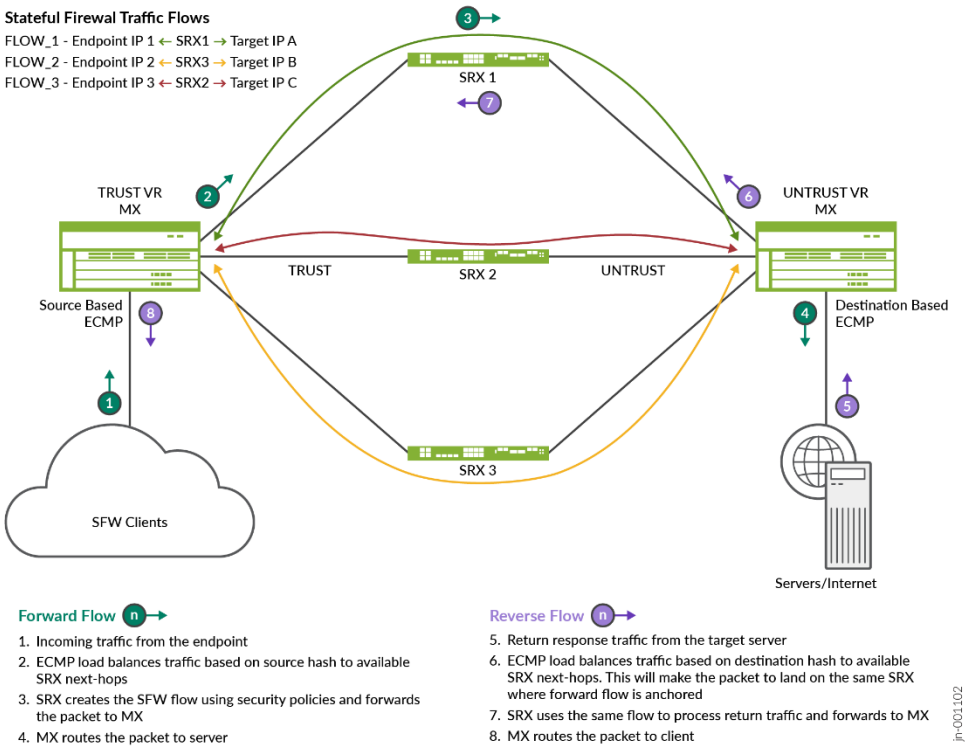
The scale-out solution is based on BGP as dynamic routing protocol. It enables all the MX Series Router and SRX Series Firewalls to learn of their surrounding networks, however, most importantly to exchange path information for the network traffic that needs to be sent from the MX Series Router across each SRX Series Firewalls to the next MX Series Router. This protocol enables the exchange of network paths for the internal/user subnets and the default/specific external network. When each SRX Series Firewalls announces what it learned from the other side, each with the same “network cost”, the load balancing can then use those routes for load balancing traffic across each SRX Series Firewalls.

Figure 11: BGP Network Announcements



The following diagram shows how traffic flows may be distributed from an MX Series Router to multiple SRX Series Firewalls using ECMP load balancing method. The SRX Series Firewalls are in a symmetric sandwich between the two MX Series Routers in the diagram, whether those MX routers are actually a single physical node configured with two routing instances (more typical) or two physical MX Series Router nodes on each side, the routing principle stays the same as if two routing nodes are used, maintaining the traffic flow distribution that is consistent in both directions.

Figure 12: Traffic Flows



The MX Series Router on the left uses TRUST-VR routing instance to forward traffic to each SRX Series Firewall.

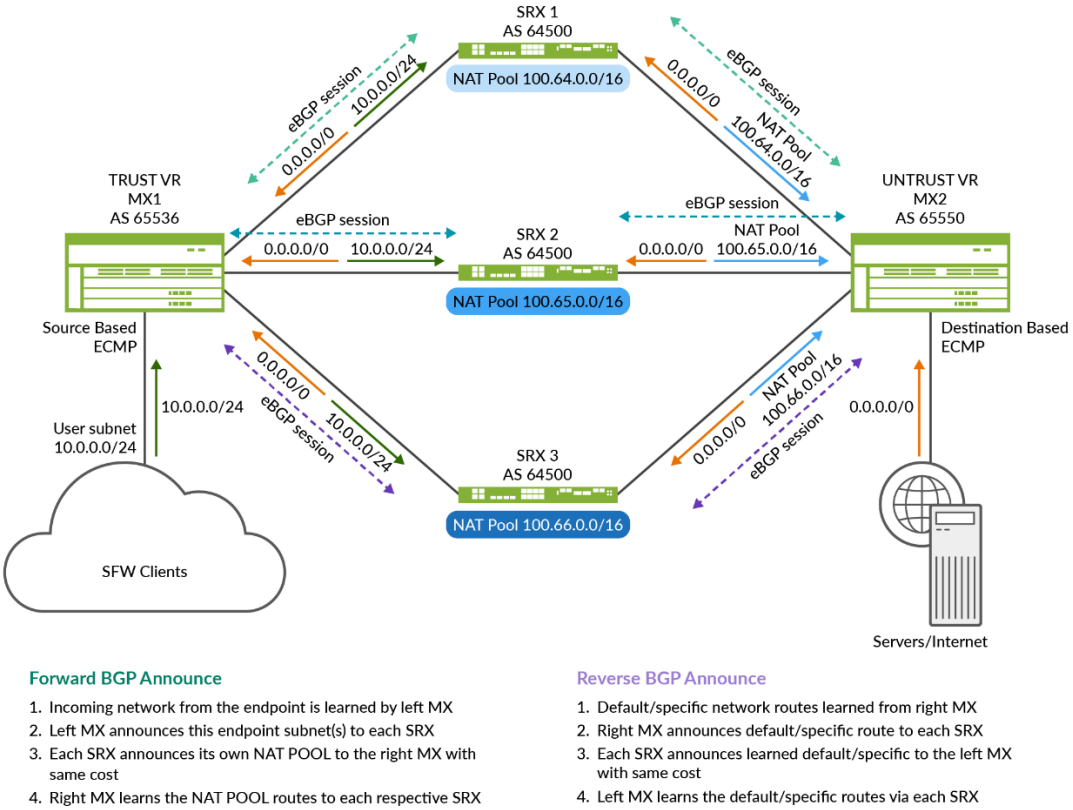
The MX Series Router on the right has used UNTRUST-VR to receive traffic from each SRX Series Firewall and forward it to the next-hop toward the target resources. The routes on each side are announced through BGP to the next hop, making its path available on each MX instance through each SRX Series Firewall (with same cost for load balancing).

Routes are announced through BGP, each MX router with their own BGP Autonomous System (AS) and peer with the SRX Series Firewall on their two sides (TRUST and UNTRUST zones in a single routing instance). The MX Series Router may peer with any other routers bringing connectivity to the clients and servers (here GW Router).

When the routes across each SRX Series Firewalls are known with similar cost, then the load balancing method can be used as explained below.

For CGNAT use case, this is very similar to SFW, however the NAT pools are exchanged on the right MX Series Router for the return traffic to flow back to the correct SRX Series Firewall:

Figure 13: Network BGP Announces with NAT Pools



Introduction to SRX Multi Node High Availability

For more information, see an extract from the public documentation on MNHA https://www.juniper.net/documentation/us/en/software/junos/high-availability/topics/topic-map/mnha-introduction.html.

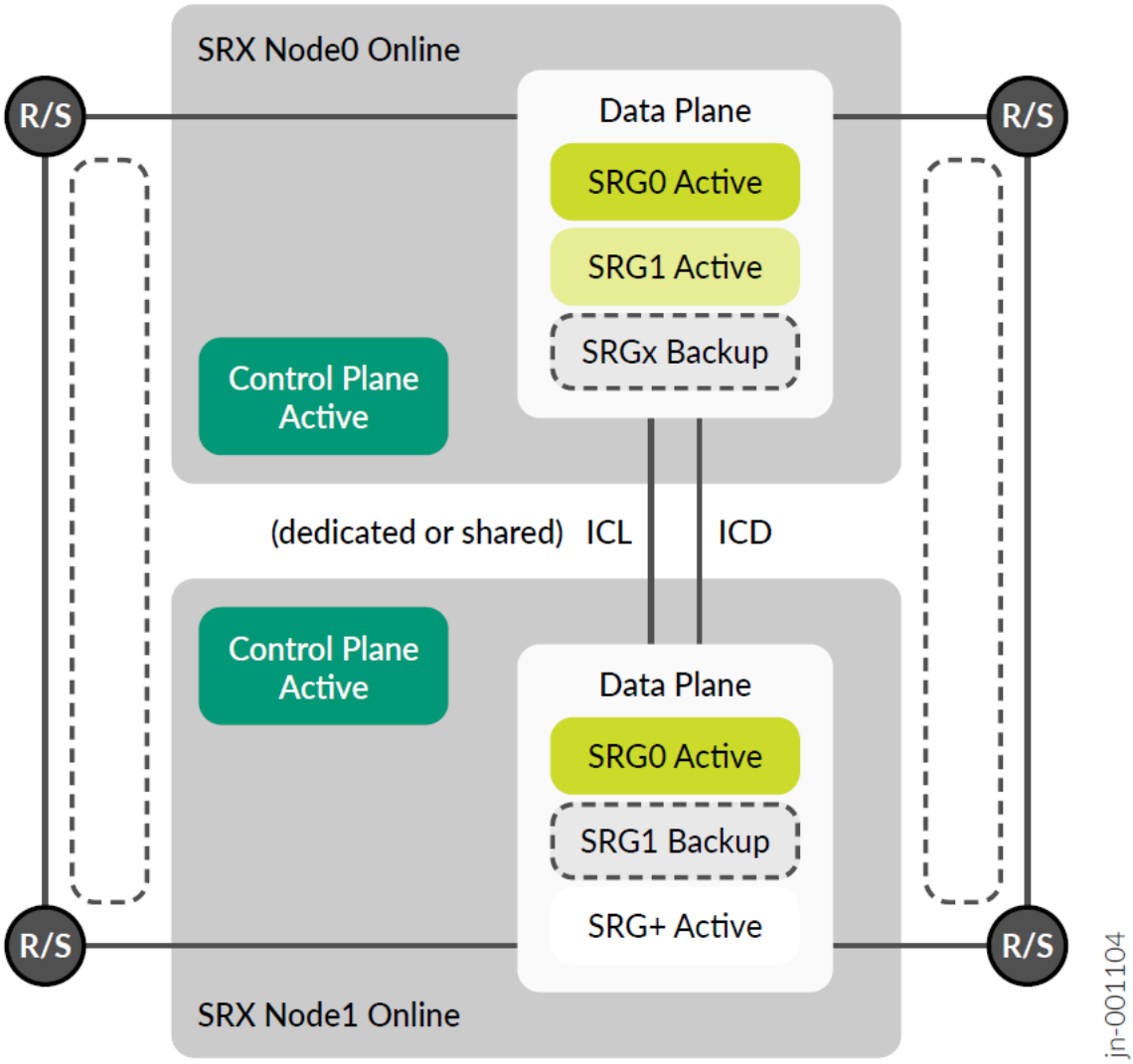
Juniper Networks® SRX Series Firewalls support a new solution, Multi Node High Availability (MNHA), to address high availability requirements for modern data centres. In this solution, both the control plane and the data plane of the participating devices (nodes) are active at the same time. Thus, the solution provides inter-chassis resiliency.

The participating devices are either co-located or physically separated across geographical areas or other locations such as different rooms or buildings. Having nodes with HA across geographical locations ensures resilient service. If a disaster affects one physical location, MNHA can fail over to a node in another physical location, thereby ensuring continuity.

In MNHA, both SRX series firewalls have an active control plane and communicate their status over an Inter Chassis Link (ICL) that can be direct or routed across the network. This allows the nodes to be geo-dispersed while synchronizing the sessions and IKE security associations. Also, they do not share a common configuration, and this enables different IP addresses settings on both SRX series firewalls. There is a commit sync mechanism that can be used for the elements of configuration to be same on both platforms.

The SRXs uses one or more SRDs for the data plane that can be either active or backup (for SRG1 and above). An exception is the SRG group 0 (zero) that is always active on both. This is a group that can be used natively by scale-out solution to load balance the traffic across both SRX Series Firewalls at the same time. However, some interest exists for the other modes where it can be Active/Backup for SRG1 and Backup/Active for SRG2. This is like always active SRG0, however can also add some routing information (like BGP as-path-prepend) under certain conditions. SRG1/+ offers more health checking of its surrounding environment that can be leveraged to make an SRGn group active/backup/ineligible.

Figure 14: Munti Node High Availability General Architecture

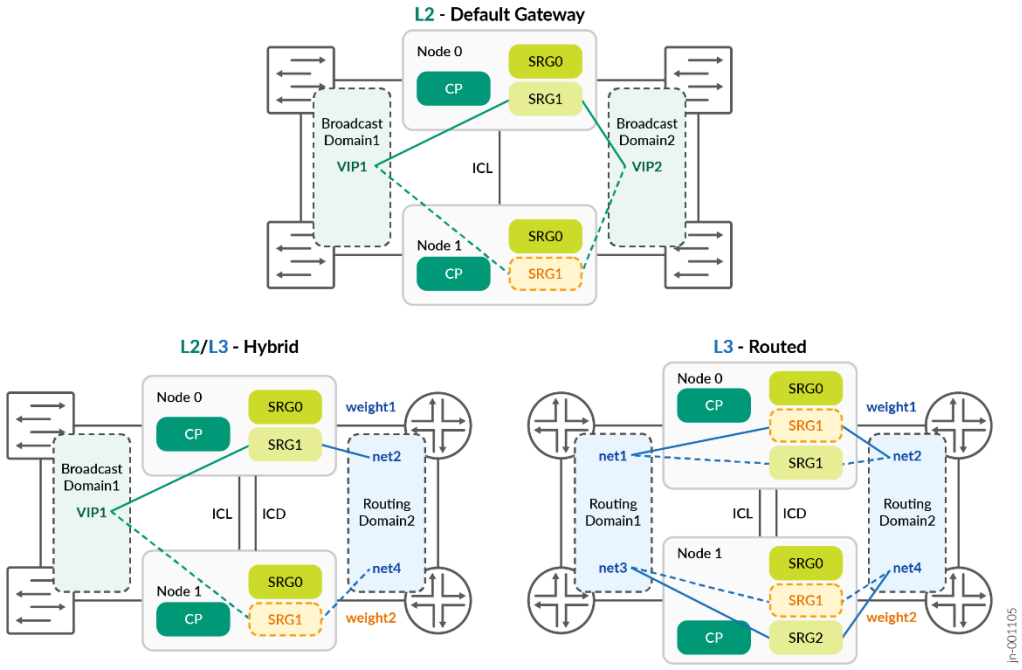


MNHA can select a network mode between the following three possibilities:

- Default Gateway or L2 mode: It uses only the same network segment at L2 on the different sides of the SRX Series Firewalls (e.g. trust/untrust) and both SRX Series Firewalls share a common IP / MAC address on each network segment. It does not mean the SRX Series Firewall is in switching mode, it does route between its interfaces, however, shares the same broadcast domain on one side with the other SRX Series Firewall, and same on the other side as well.
- Hybrid mode or mix of L2 and L3: It uses an L2 and IP address on one side of the SRX Series Firewall (e.g. trust) and routing on the other side (e.g. untrust) then having different IP subnets on the second side.

- Routing mode or L3: This is the architecture used for this JVD where each side of the SRX Series Firewall is using different IP address, even between the SRX Series Firewalls (no common IP subnet) and all communication with the rest of the network is done through routing. This mode is perfect for scale-out communication using BGP with the MX Series Router.

Figure 15: Multi Node High Availability Network Modes

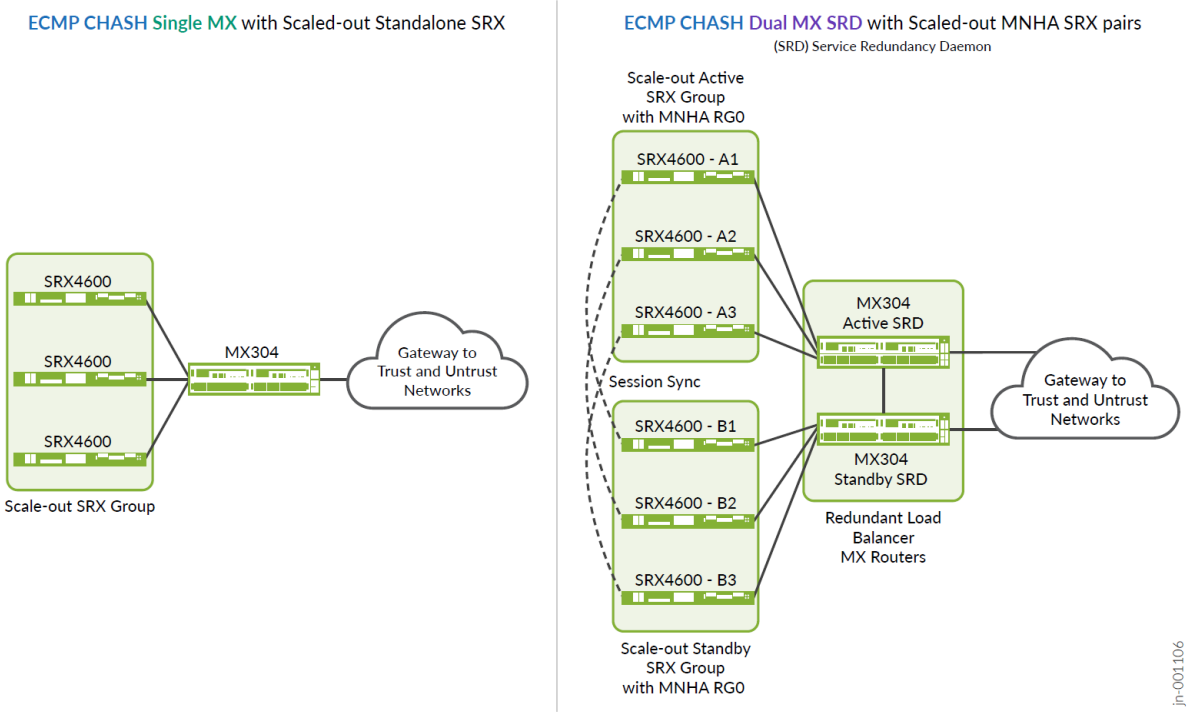


Whether using SRG0 Active/Active, or SRG1 Active/Backup (single one active at a time), or a combination of SRG1 Active/Backup and SRG2 Backup/Active, this simply uses one or two SRX Series Firewalls in a cluster at the same time.

ECMP/Consistent Hashing (CHASH) Load Balancing Overview

This feature relates to topology 1 (single MX Series Router, scale-out SRXs) and topology 2 (dual A/P MX Series Router and scale-out SRX MNHA pairs).

Figure 16: : Topologies 1 and 2 - ECMP CHASH



ECMP/Consistent Hashing (CHASH) in MX Router:

ECMP is a network routing strategy that transmits traffic of the same session, or flow — that is, traffic with the same source and destination across multiple paths of equal cost. It is a mechanism that allows to load balance traffic and increase bandwidth (by fully utilizing) otherwise unused bandwidth on links to the same destination.

When forwarding a packet, the routing technology must decide which next-hop path to use. The device considers the packet header fields that identify a flow. When ECMP is used, next-hop paths of equal cost are identified based on routing metric calculations and hash algorithms. That is, routes of equal cost have the same preference and metric values, and the same cost to the network. The ECMP process identifies a set of routers, each of which is a legitimate equal cost next-hop towards the destination. The routes that are identified are referred to as an ECMP set. An ECMP set is formed when the routing table contains multiple next-hop addresses for the same destination with equal cost (routes of equal cost have same preference and metric values). If there is an ECMP set for the active route, Junos OS uses a hash algorithm to choose one of the next-hop addresses in the ECMP set to install in the forwarding table. You can configure Junos OS so that multiple next-hop entries in an ECMP set are installed in the forwarding table. On Juniper Networks devices, per-packet load balancing is performed to spread traffic across multiple paths between routing devices.

The following example is of learned routes and forwarding table for the same destination (assuming traffic target is within 100.64.0.0/16 and SRX BGP peers are 10.1.1.0, 10.1.1.8 and 10.1.1.16):

```
jcluser@mx-01> show route 100.64.0.0/16
trust-vr.inet.0: 30 destinations, 33 routes (30 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
100.64.1.0/16        *[BGP/170] 4d 04:52:53, MED 10, localpref 100
                      AS path: 64500 64500 I, validation-state: unverified
                        to 10.1.1.0 via ae1.0      ## learning routes from BGP peer SRX1
                      >  to 10.1.1.8 via ae2.0      ## learning routes from BGP peer SRX2
                        to 10.1.1.16 via ae3.0     ## learning routes from BGP peer SRX3
jcluser@mx-0> show route forwarding-table destination 100.64.1.0/16 table trust-vr
Routing table: trust-vr.inet
Internet:
Destination        Type RtRef Next hop           Type Index    NhRef Netif
100.64.1.0/16       user     0                    ulst  1048580     2
                             10.1.1.0            ucst      801     4 ae1.0      ## to SRX1
                             10.1.1.8            ucst      798     5 ae2.0      ## to SRX2
                             10.1.1.16           ucst      799     5 ae3.0      ## to SRX3
```

With scale-out architecture where stateful security devices are connected, maintaining symmetricity of the flows in the security devices is the primary objective. The symmetricity means traffic from a subscriber (user) and to the subscriber must always reach the same server (which maintains the subscriber state). To reach the same server, the traffic must be hashed onto the same link towards that server for traffic in both directions.

A subscriber is identified by the source IP address in the upstream direction (client to server) and by the destination IP address in the downstream direction (server to client). The MX Series Routers do symmetric hashing i.e. for a given (sip, dip) tuple, same hash is calculated irrespective of the direction of the flow i.e. even if sip and dip are swapped. However, the requirement is that all flows from a subscriber reach the same SRX Series Firewall so you need to hash only on source IP address (and not destination IP address) in one direction and vice versa in the reverse direction.

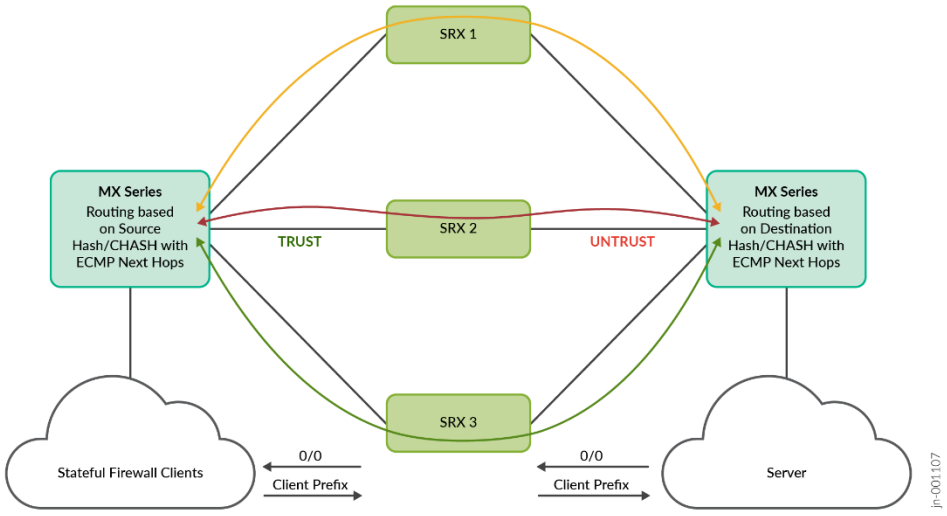
By default, when a failure occurs in one or more paths, the hashing algorithm recalculates the next hop for all paths, typically resulting in redistribution of all flows. Consistent load balancing enables you to override this behavior so that only flows for inactive links are redirected. All existing active flows are maintained without disruption. In such an environment, the redistribution of all flows when a link fails potentially results in significant traffic loss or a loss of service to SRX Series Firewall whose links remain active. However, consistent load balancing maintains all active links and remaps only those flows affected by one or more link failures. This feature ensures that flows connected to links that remain active continue to remain uninterrupted.

This feature applies to topologies where members of an ECMP group are external BGP neighbors in a single-hop BGP session. Consistent load balancing does not apply when you add a new ECMP path or modify an existing path in any way. New SRX add design is implemented recently where you can add SRX Series Firewall gracefully with an intent of equal redistribution from each active SRX Series Firewall, hence causing minimal impact to the existing ECMP flows. For example, if there are four active SRX Series Firewalls carrying 25% of total flows on each link and a 5th SRX Series Firewalls (previously unseen) is added, 5% of flows from each existing SRX Series Firewalls moves to the new SRX Series Firewalls. Hence making 20% of flow re-distribution from existing four SRX Series Firewalls to the new one.

The following information shares details for each step of route exchange between MX Series Router and SRXs, traffic flows, for each use case.

ECMP/CHASH in Topology 1 (Single MX, scale-out SRXs) for SFW:

Figure 17: Topology 1 - ECMP CHASH - SFW Use Case

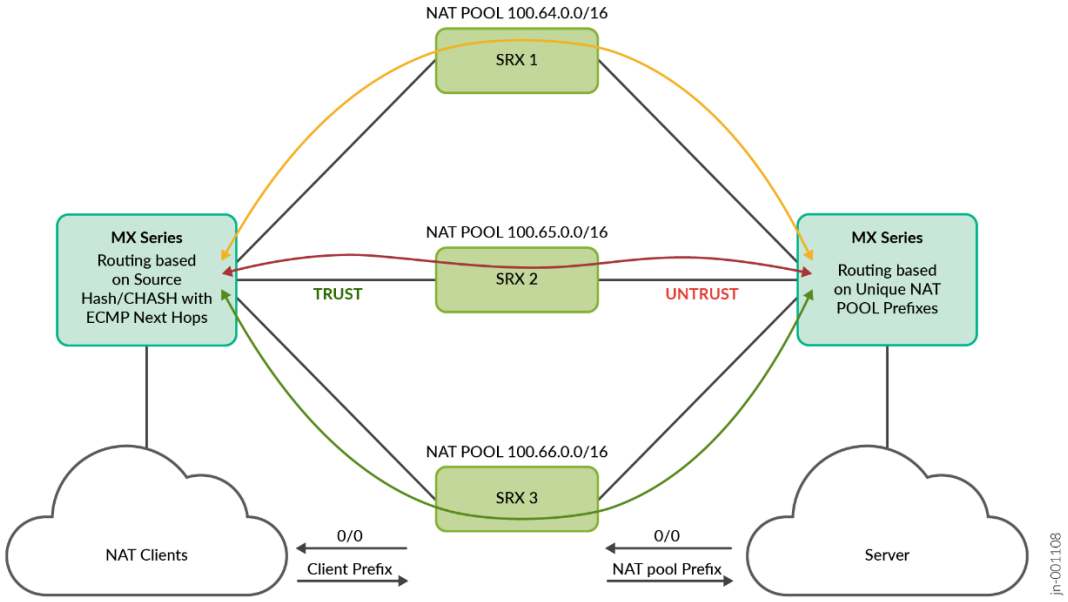


- SRX Series Firewalls are deployed in a standalone scaled out devices to single MX Series Router.
- Links between MX Series Routers and all SRX Series Firewalls are configured with two eBGP sessions. One for TRUST and one for UNTRUST.
- The load balancing policy with source-hash for route 0/0 is configured in the forwarding table.
- The load balancing policy with destination-hash for client prefix routes (users) is configured in the forwarding table.

- The default 0/0 route is received by all the SRX Series Firewalls on UNTRUST side and advertised using eBGP to MX Series Router on the TRUST side. The MX Series Router imports this route on the TRUST instance using load balancing CHASH policy.
- Client prefix route is received by all the SRX Series Firewalls on TRUST side and advertised using eBGP to MX Series Router on the UNTRUST side. The MX Series Router imports this route on the UNTRUST instance using load balancing CHASH policy.
- The MX Series Router on the TRUST side has all the ECMP routes for 0/0 route.
- The MX Series Router on the UNTRUST side has all the ECMP routes for the client prefix routes.
- Forward traffic flow from client to server reaches MX Series Router on TRUST instance and hits 0/0 route and takes any one ECMP next-hop to SRX series firewall based on the calculated source IP based hash value.
- The SRX Series Firewalls creates an SFW flow session and routes the packet to MX Series Router on the UNTRUST direction towards the server.
- Reverse traffic flow from server to client reaches MX Series Router on UNTRUST instance and hits client prefix route and takes the same ECMP next hop based on the calculated destination IP based hash value.
- Since the five tuples of the SFW sessions do not change, calculated hash value remains the same and takes the same ECMP next hop/SRX Series Firewalls on the forward and reverse flow. This makes sure symmetricity is maintained in the SRX Series Firewalls.
- When any SRX Series Firewall goes down, CHASH on the MX Series Router ensures that the sessions on the other SRX Series Firewalls are not disturbed and only sessions on the down SRX Series Firewalls are redistributed.

ECMP/CHASH in Topology 1 (Single MX, scale-out SRXs) for CGNAT:

Figure 18: Topology 1 - ECMP CHASH - CGNAT Use Case

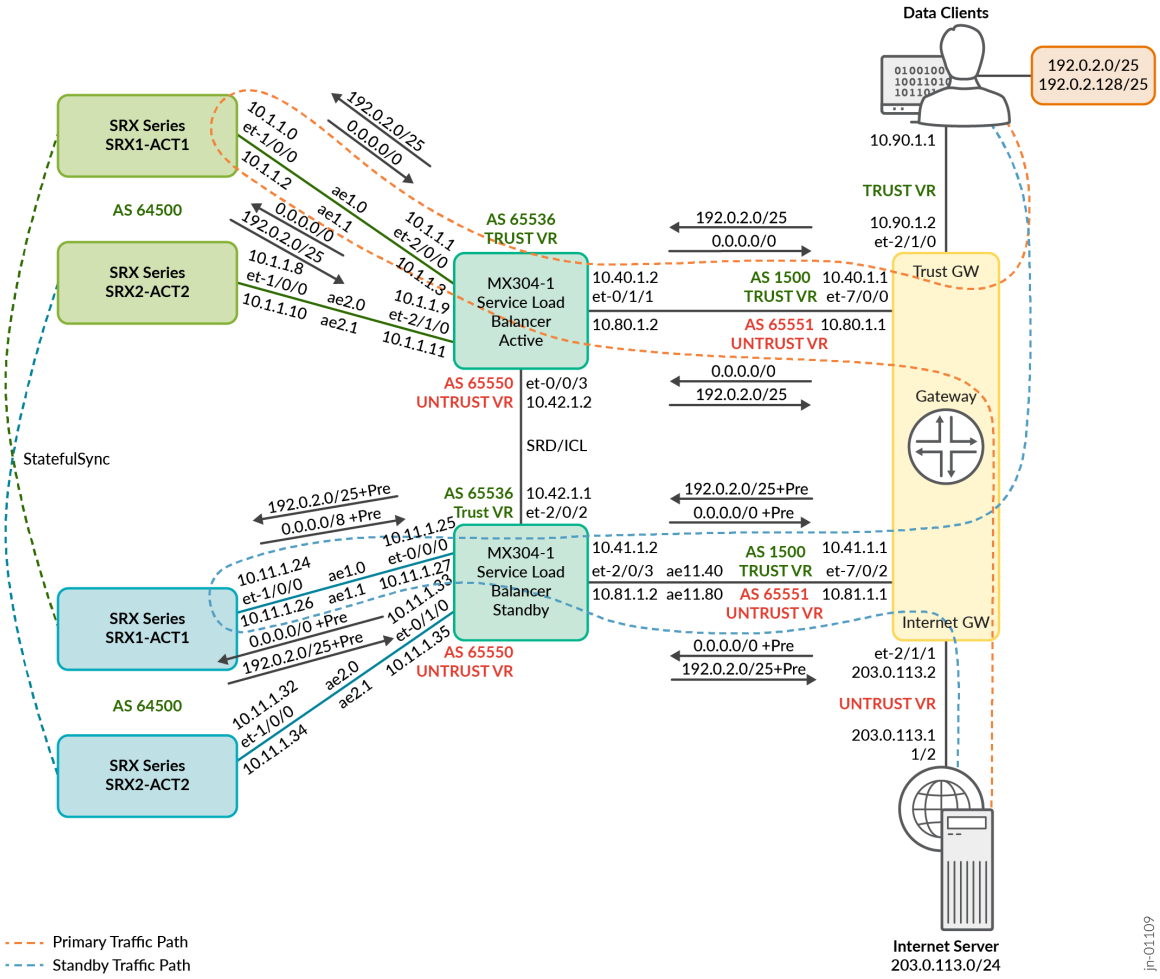


- The SRX Series Firewalls are deployed in a standalone scaled out devices to a single MX Series Router.
- Links between the MX Series Router and SRX Series Firewalls are configured with two eBGP sessions. One for TRUST and one for UNTRUST.
- Unique NAT pool IP address ranges are allocated per SRX Series Firewalls.
- The load balancing policy with source-hash for route 0/0 is configured in the forwarding table.
- 0/0 route is received by the SRX Series Firewalls on the Untrust side and is advertised using eBGP to MX Series Router on the TRUST side. The MX Series Router imports this route on the TRUST instance using load balancing CHASH policy.
- Client prefix route is received by the SRX Series Firewalls on the TRUST side and NAT pool route prefix is advertised using eBGP to MX Series Router on the UNTRUST side.
- The MX Series Router on the TRUST side has an ECMP route for 0/0 route.
- The MX Series Router on the UNTRUST side has a unique route for the NAT pool route prefix.

- The forward traffic flow from client to server reaches the MX Series Router on TRUST instance and hits 0/0 route. It takes any one ECMP next-hop to SRX Series Firewalls based on the calculated source IP based hash value.
- The SRX Series Firewalls creates an NAT flow session and routes the packet to MX Series Router on the UNTRUST direction towards the server.
- Reverse traffic flow from server to client reaches MX Series Router on UNTRUST instance and hits unique NAT pool prefix route and takes the same SRX Series Firewalls where forward flow is anchored. This makes sure symmetricity is maintained in the SRX devices.
- When any SRX Series Firewall goes down, CHASH on the MX Series Router ensures that the sessions on the other SRX Series Firewalls are not disturbed and only sessions on the down SRX Series Firewalls are redistributed.

ECMP/CHASH in Topology 2 (Dual MX, SRX MNHA Pairs) for SFW:

Figure 19: Topology 2 - ECMP CHASH - SFW Use Case



- When the SRX Series Firewalls are deployed in pair with MNHA, session syncs both ways depending on where the traffic is received.
- The MX Series Router pair is configured with SRD redundancy for user management of the MX HA pair.
- The MX Series Router pair monitor links towards Trust GW / Internet GW router and links between the MX Series Router to the SRX Series Firewalls. SRD triggers automatic switch over to another MX Series Router if any of this link fails. It can also failover when MX304-1 completely goes down. The MX Series Routers have 4x100G interface connected to the SRX4600 devices as an AE bundle and

contain three VLANs (trust, untrust and HA management).MX304-1 remains primary ECMP path and MX304-2 standby ECMP path.

- SRD is used for MX Series Router redundancy and controls the MX master ship state transition. It also installs a signal route on the master MX Series Router which is used for route advertisement with preference.
- MX304-1 advertises routes as it is, whereas MX304-2 standby advertises routes with as-path-prepend.
- Interfaces on MX304-1 towards SRX Series Firewalls and MX304-2 towards SRX Series Firewalls need to be provisioned using similar interface numbering with similar I/O card. This helps in maintaining the same unilist next-hop ordering on both the MX304-1 and MX304-2 routers. RPD decides unilist next-hop ordering based on the interface ifl index number (Ascending order of interface ifl numbers).
- Since unilist next-hop ordering is same in both MX Series Router, post any MX Series Router switchover, there is not going to be any issue with hash (source or destination).
- If any failure is detected by an active MX Series Router (SRD), the failover to the other MX Series Router. This implies that all traffic reaches this second MX Series Router (Second MX Series Router has taken ownership of the SRX Series Firewalls and announced around the routes to itself). It also implies that traffic to SRX Series Firewalls connected to MX304-1 is sent to SRXs connected to MX304-2. This is a complete failover of the top architecture to the bottom one.

The following MX Series Router configuration shows how the SRD process monitors events to decide any release or acquisition of mastership. On the SRD process side, the relevant configuration contains:

```
event-options {
    redundancy-event 1_MSHIP_ACQUIRE_EVENT {
        monitor {
            peer {                      ### Monitored membership released from MX peer
                mastership-release;
            }
        }
    }
    redundancy-event 1_MSHIP_RELEASE_EVENT {
        monitor {
            link-down {                ### Monitored interfaces when link is down
                ae1;
                ae1.0;
                ae1.1;
                ...
                ...
```

```
                ae11.80;
            }
            process {                          ### Monitored process restarting
                routing {
                    restart;
                }
            }
            peer {                                    ### Monitored membership acquisition from
MX peer
                mastership-acquire;
            }
        }
    }
}
services {
    redundancy-set {
        1 {
            redundancy-group 1;
            redundancy-policy [ 1_ACQU_MSHIP_POL 1_RELS_MSHIP_POL ];
        }
    }
}
policy-options {
    redundancy-policy 1_ACQU_MSHIP_POL {
        redundancy-events 1_MSHIP_ACQUIRE_EVENT;
        then {
            acquire-mastership;
            add-static-route 10.3.0.3/32 {### Adds this route when acquiring mastership
                receive;
                routing-instance SRD;
            }
        }
    }
    redundancy-policy 1_RELS_MSHIP_POL {
        redundancy-events 1_MSHIP_RELEASE_EVENT;
        then {
            release-mastership;
            delete-static-route 10.3.0.3/32 {### Removes this route when releasing mastership
                routing-instance SRD;
            }
        }
```

```
    }
}
```

On the routing side, the SRD configuration looks for the existence of specific route and then announces the default route conditionally:

```
interfaces {
    ae10 {
        description To-GW;
        vlan-tagging;
        unit 40 {
            description "GI-POC MX-GW TRUST_VR_v4";
            vlan-id 40;
            family inet {
                address 10.40.1.2/24 {
                    vrrp-group 40 {
                        virtual-address 10.40.1.1;
                        ...
                        ...
                        track {
                            route 10.3.0.3/32 routing-instance SRD priority-cost 30;
                        }
                    }
                }
            }
        }
        unit 80 {
            description "GI-POC MX-GW UNTRUST_VR_v4";
            vlan-id 80;
            family inet {
                address 10.80.1.2/24 {
                    vrrp-group 80 {
                        virtual-address 10.80.1.1;
                        ...
                        ...
                        track {
                            route 10.3.0.3/32 routing-instance SRD priority-cost 30;
                        }
                    }
                }
            }
        }
```

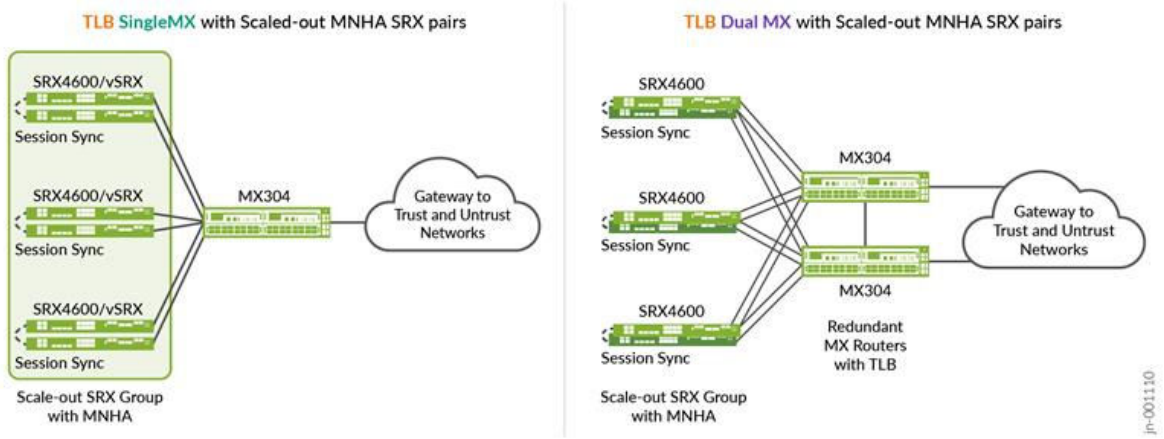
```
    }
}
policy-options {
    condition 1_ROUTE_EXISTS {
        if-route-exists {                              ### If this route exists….
            10.3.0.3/32;
            table SRD.inet.0;
        }
    }
    policy-statement MX-to-MX-1_trust_export {
        term 1 {
            from {
                protocol bgp;
                route-filter 0.0.0.0/0 exact;  ### Then announces default route to self on trust-
vr
                condition 1_ROUTE_EXISTS;
            }
            then {
                local-preference 200;
                next-hop self;
                accept;
            }
        }
        term 2 {
            then reject;
        }
    }
    policy-statement MX-to-MX-1_untrust_export {
        term 1 {
            from {
                protocol bgp;          ### Then announces clients route to self on untrust-vr
                prefix-list-filter GI-FW_clients_v4 exact;
                condition 1_ROUTE_EXISTS;
            }
            then {
                local-preference 200;
                next-hop self;
                accept;
            }
        }
        term 2 {
            then reject;
        }
```

```
    }
}
routing-instances {
    1_TRUST_VR {
        instance-type virtual-router;
        protocols {
            bgp {
                group MX-TO-MX-IBGP {
                    type internal;
                    export MX-to-MX-1_trust_export;     ### Export conditional route to external
GW
                    ...
                }
            }
        }
        ...
        ...
        interface ae10.40;
    }
    1_UNTRUST_VR {
        instance-type virtual-router;
        protocols {
            bgp {
                group MX-TO-MX-IBGP {
                    type internal;
                    export MX-to-MX-1_untrust_export;   ### Export conditional route to external
GW
                    ...
                }
            }
        }
        ...
        ...
        interface ae10.80;
    }
    SRD {
        instance-type virtual-router;
        ...
    }
}
```

Traffic Load Balancer Overview

This feature relates to topology 3 (single MX Series Router, scale-out SRX MNHA pairs) and topology 4 (dual MX Series Routers and scale-out SRX MNHA pairs).

Figure 20: Topology 3 and 4 - TLB - SFW Use Case



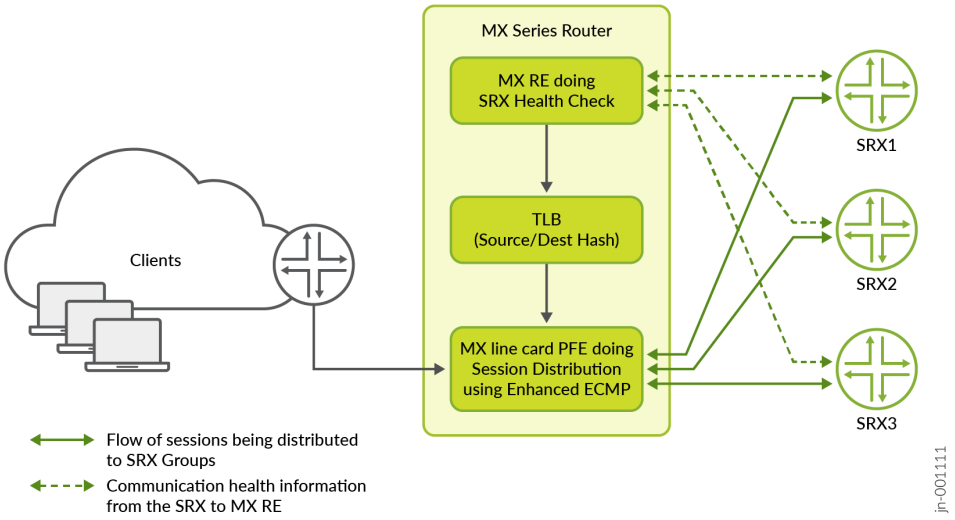
Traffic Load Balancer in MX Router:

Traffic Load Balancer (TLB) functionality provides stateless translated or non-translated traffic load balancer, as an inline PFE service in the MX Series Routers. Load balancing in this context is a method where incoming transit traffic is distributed across configured servers that are in service. This is a stateless load balancer, as there is no state created for any connection, and so there are no scaling limitations. Throughput could be close to line rate. TLB has two modes of load balancing i.e., translated (L3) and non-translated Direct Server Return (L3).

For the scale-out solution, the TLB mode non-translated Direct Server Return (L3) is used. As part of TLB configuration, there is a list of available SRX Series Firewalls addresses and the MX PFE programs a selector table based on this SRX Series Firewalls. TLB does a health check (ICMP usually however it can do HTTP, UDP, and TCP checks) for each of the SRX Series Firewalls individually. TLB health check is done using MX Series Router routing engine. If the SRX Series Firewalls pass the health check, TLB installs a specific IP address route or wild card IP address (TLB config option) route in the routing table with next-hop as composite next-hop. Composite next-hop in the PFE is programmed with all the available SRX Series Firewalls in the selector table. Filter based forwarding is used to push the "Client to Server" traffic to the TLB where it hits the TLB installed specific IP address route or wild card IP address

route to get the traffic sprayed across the available SRX Series Firewalls with source or destination hash. "Server to Client" is directly routed back to client instead of going through the TLB.

Figure 21: TLB Work in RE and PFE



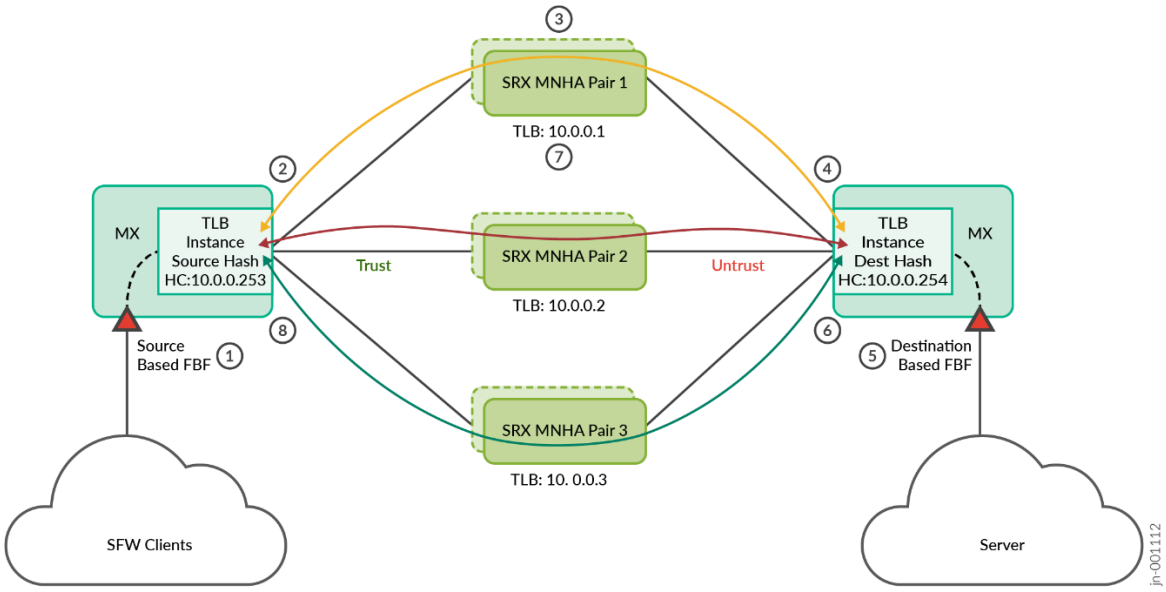
TLB is used in Junos and MX routers family for few years now (as early as Junos 16.1R6) and you have been using it successfully on large server farms with around 20,000 servers.

TLB uses the control part and the health check on MS-MPC or MX-SPC3 service cards on MX240/480/960 and MX2000 chassis before data plane or PFE is already on the line cards. It is not running on the RE as it is implemented on MX304/MX10000 chassis.

For more information see, https://www.juniper.net/documentation/us/en/software/junos/interfaces-next-gen-services/interfaces-adaptive-services/topics/concept/tdf-tlb-overview.html

Using TLB in the MX Router for the Scale-Out SRX Solution with SFW:

Figure 22: Topology 3 - Scale-Out SFW with TLB



Forward Flow

1. Filter based Forwarding based on source prefix to TLB Instance
2. TLB load balances traffic based on source hash to available SRX next-hops
3. SRX creates the SFW flow using and forwards the packet to MX
4. MX routes the packet to server

Reverse Flow

5. Filter based Forwarding based on destination prefix to TLB Instance
6. TLB load balances traffic based on destination hash to available SRX next-hops. This will make the packet to land on the same SRX where forward flow is anchored
7. SRX uses the same flow to process return traffic and forwards to MX
8. MX routes the packet to client

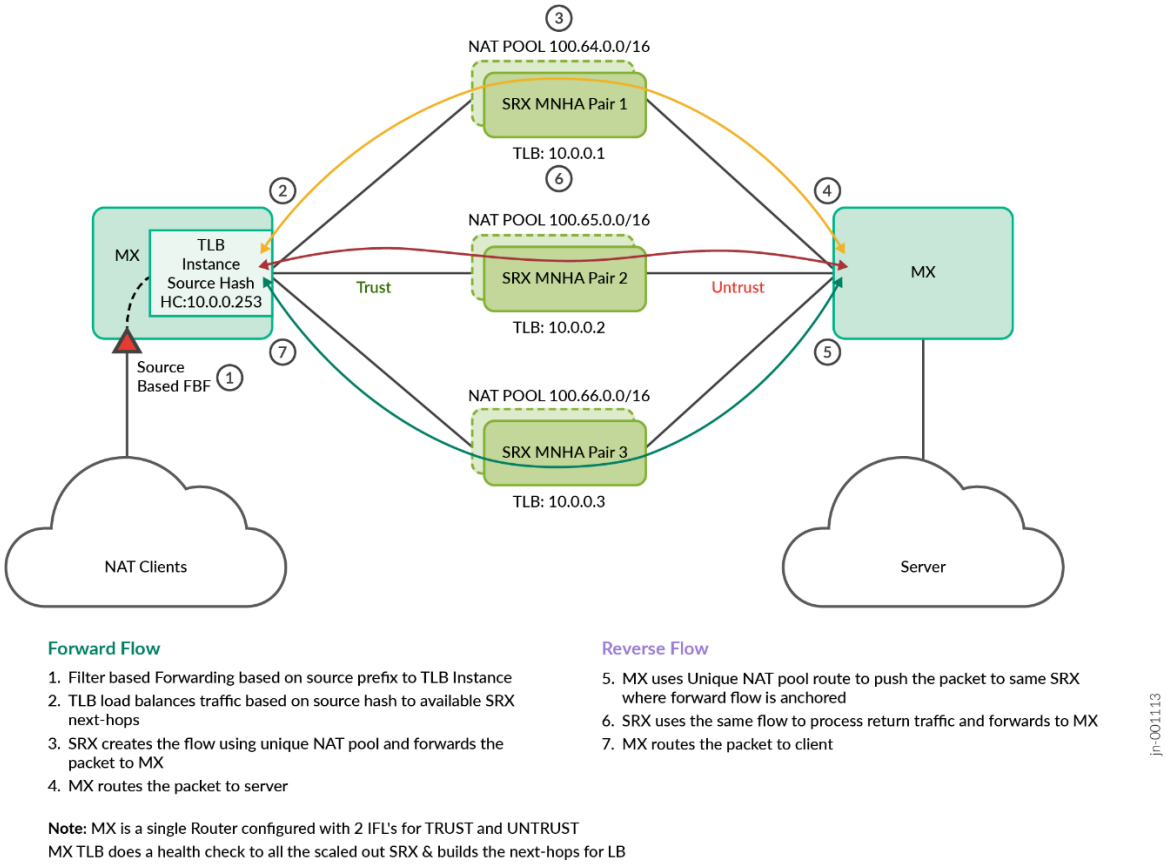
Note: MX is a single Router configured with 2 IFL's for TRUST and UNTRUST
MX TLB does a health check to all the scaled out SRX & builds the next-hops for LB

- All the SRX Series Firewalls are configured with BGP to establish an eBGP peering sessions with MX Series Router nodes.
- The MX Series Router is configured with TLB on the Trust routing instance to do the load balancing of data traffic coming from client-side gateway router towards scaled out SRX Series Firewalls.
- All the scale-out SRX Series Firewalls connected to MX Series Router are configured with unique IP address (for example, loopback) which is used by MX TLB to do the health check and build up the selector table in the PFE. PFE uses this selector table to load balance the packet across the available next hops. This health check is reachable through BGP connection.
- Filter based forwarding based on source IP address match is used in MX Series Router to push SFW specific traffic to the TLB trust forwarding instance.

- TLB forwarding instance has a default route with next-hop as list of SRX Series Firewalls. TLB installs this default route when its health check passes with at least one SRX Series Firewalls.
- TLB does source based hash load balancing across all the available SRX next-hop devices.
- Load balanced SFW data sessions are anchored on any available SRX Series Firewalls and SFW flow gets created. Then it is routed to reach the server through MX Series Router over Untrust routing instance.
- For the return traffic coming from server to client direction on the MX Untrust routing instance, another TLB instance is configured on MX Untrust routing instance to do the load balancing back to the same SRX Series Firewalls.
- Filter based forwarding of destination IP address match is used in MX Series Router to push SFW specific traffic to the TLB UNTRUST forwarding instance.
- TLB forwarding instance has a default route with next-hop as list of SRX Series Firewalls. TLB installs this default route when its health check passes with at least one SRX Series Firewalls.
- TLB does destination-based hash load balancing across all the available SRX next-hop devices.
- Load balanced SFW data sessions are load balanced to the same SRX Series Firewalls on the return direction and uses the same flow to reach the client through MX Series Router over TRUST routing instance.

Using TLB in the MX Router for the Scale-Out SRX Solution with CGNAT:

Figure 23: Topology 3 - Scale-Out CGNAT with TLB



- All the scale-out SRX Series Firewalls connected to MX Series Routers are configured with BGP connections.
- Each scaled out SRX Series Firewall needs to have a unique NAT pool range, and this must be advertised towards the MX Untrust direction. (This is the main difference with SFW use case, as it needs to announce the NAT pools)
- The MX Series Router is configured with TLB on the Trust routing instance to do the load balancing of data traffic coming from client-side gateway router towards scaled out SRX Series Firewalls.
- All the scale-out SRX Series Firewalls connected to the MX Series Router are configured with unique IP address, which is used by MX TLB to do the health check and build the selector table in the PFE. PFE uses this selector table to load balance the packet across the available next hops. This health check is reachable through BGP connection.

- The filter-based forwarding on source IP address match is used in the MX Series Router to push the NAT specific traffic to the TLB trust forwarding instance.
- The TLB forwarding instance has a default route with next-hop as list of SRX Series Firewalls. TLB installs this default route when its health check passes with at least one SRX Series Firewalls.
- TLB does source-based hash load balancing across all the available SRX next-hop devices.
- Load balanced NAT data sessions are anchored on any available SRX Series Firewalls and NAT flow gets created. Then it is routed to reach the server through MX Series Router over UNTRUST routing instance.
- For the return traffic coming from server to client direction on the MX Untrust routing instance, Unique NAT pool routes are used to route the traffic to the same SRX devices.
- The SRX Series Firewalls use same NAT flow to process the return traffic and route the packet towards MX Series Router on the TRUST direction. The MX Series Router routes the packet back to the client.

Configuration Examples for ECMP CHASH

The following sample configurations are proposed to understand the elements making this solution work, including configurations for both MX Series Router and some SRX Series Firewalls. It contains a lot of repetitive statements. It shows Junos hierarchical view.

Source-hash for forward flow and destination-hash for reverse flow is common for all ECMP based solutions or TLB based solutions. Consistent hash (CHASH) is used during any next-hop failure where it helps an existing session on an active next-hop to remain undisturbed, while sessions on down next-hop is redistributed over other active next-hop. This CHASH behavior is pre-built in the TLB solution. However, in ECMP based solution you must configure this CHASH configuration explicitly using BGP import policy.

The following sample MX Series Router configuration is for ECMP load balancing using source and destination hash:

```
policy-options {
    prefix-list clients_v4 {                    ### clients subnet(s)
        192.0.2.0/25;
    }
    policy-statement pfe_lb_hash {
        term source_hash {
            from {
```

```
                route-filter 0.0.0.0/0 exact;
            }
            then {
                load-balance source-ip-only;       ### when 0/0, then LB per source-ip
                accept;
            }
        }
        term dest_hash {
            from {
                prefix-list-filter clients_v4 exact;
            }
            then {
                load-balance destination-ip-only;###when clients, then LB per dest-ip
                accept;
            }
        }
        term ALL-ELSE {
            then {
                load-balance per-packet;                ### for anything else
                accept;
            }
        }
    }
}
routing-options {
    forwarding-table {
        export pfe_lb_hash;
    }
}
```

The following MX Series Router configuration is an example for specific forward and return traffic with CHASH:

```
policy-options {
    policy-statement pfe_consistent_hash {      ### Load Balancing mechanism
        from {
            route-filter 0.0.0.0/0 exact;
        }
        then {
            load-balance consistent-hash;
            accept;
        }
```

```
    }
    policy-statement pfe_sfw_return_consistent_hash {### Load Balancing mechanism
        from {
            prefix-list-filter clients_v4 exact;
        }
        then {
            load-balance consistent-hash;
            accept;
        }
    }
    policy-statement trust-to-untrust-export { ### Export static + learned BGP routes
        term 1 {
            from protocol [ bgp static ];
            then {
                next-hop self;
                accept;
            }
        }
        term 2 {
            then reject;
        }
    }
    policy-statement untrust-to-trust-export { ###  Export static + learned BGP routes
        term 1 {
            from protocol [ bgp static ];
            then {
                next-hop self;
                accept;
            }
        }
        term 2 {
            then reject;
        }
    }
    policy-statement client_sfw_and_nat_pool_prefix_export {
        term 1 {
            from {
                protocol bgp;
                route-filter 192.0.2.0/25 exact; ### clients subnet(s)
                route-filter 100.64.0.0/16 exact;   ### NATPool SRX1 or SRXMNHA pair1
                route-filter 100.65.0.0/16 exact;   ### NATPool SRX2 or SRXMNHA pair2
                route-filter 100.66.0.0/16 exact;   ### NATPool SRX3 or SRXMNHA pair3
                ...
```

```
            }
            then {
                next-hop self;
                accept;
            }
        }
        term 2 {
            then reject;
        }
    }
}
```

The following MX Series Router configuration is for the routing instance and BGP peering with default GW and the SRX Series Firewall:

```
routing-instances {
    TRUST_VR {
        instance-type virtual-router;
        routing-options {
            autonomous-system 65536;
            static {
                route 192.0.2.0/25 next-hop 10.40.1.1; ### Internal route to    clients
            }                              ### Or internal BGP peering (not shown)
        }
        protocols {
            bgp {
                group MX-TO-SRXS {              ### BGP Peering with all SRX (trust)
                    type external;
                    import pfe_consistent_hash;         ### apply LB CHASH for clients
                    export trust-to-untrust-export; ### Export learned routes to peers
                    peer-as 64500;
                    local-as 65536;
                    multipath;
                    bfd-liveness-detection {
                        minimum-interval 300;
                        minimum-receive-interval 300;
                        multiplier 3;
                    }
                    neighbor 10.1.1.0;                  ### PEERING WITH SRX1
                    neighbor 10.1.1.8;                  ### PEERING WITH SRX2
                    ...                                         ### ANY OTHER SRX/VSRX
                }
```

```
            }
        }
        interface ae1.0;
        interface ae2.20;
        ...
    }
    UNTRUST_VR {
        instance-type virtual-router;
        routing-options {
            autonomous-system 65550;
            static {
                route 0.0.0.0/0 next-hop 10.80.1.1; ### EITHER Ext default GW router
            }
        }
        protocols {
            bgp {
                group MX-TO-GATEWAY {            ### OR External default GW router
                    type external;
                    export client_sfw_and_nat_pool_prefix_export;      ### Export clients/NAT
routes to GW
                    neighbor 10.80.1.1;                                 ### Peering with GW
                    peer-as 65551;
                    local-as 65550;
                    ...
                }
                group MX-TO-SRXS {              ### BGP Peering with all SRX (untrust)
                    type external;
                    import pfe_sfw_return_consistent_hash;              ### Apply LB per clients
dest IP
                    export untrust-to-trust-export;                     ### Export learned
routes to peers
                    peer-as 64500;
                    local-as 65550;
                    multipath;
                    bfd-liveness-detection {
                        minimum-interval 300;
                        minimum-receive-interval 300;
                        multiplier 3;
                    }
                    neighbor 10.1.1.2;                  ### PEERING WITH SRX1
                    neighbor 10.1.1.10;                         ### PEERING WITH SRX2
                    ...                                         ### ANY OTHER SRX/VSRX
            }
```

```
        }
        interface ae...;
        ...
    }
}
```

The following is the sample SRX1 configuration for SFW and CGNAT:

```
policy-options {
    policy-statement trust_export_policy {
        term 1 {
            from {
                protocol bgp;
                route-filter 0.0.0.0/0 exact;          ### SRX announce 0/0 to MX
            }
            then {
                next-hop self;
                accept;
            }
        }
        term 2 {
            then reject;
        }
    }
    policy-statement untrust_export_policy {
        term 1 {
            from {
                protocol bgp;
                route-filter 192.0.2.0/25 orlonger; ### SRX announce clients to MX
            }
            then accept;
        }
        term 2 {
            from {
                protocol static;
                route-filter 100.64.0.0/16 orlonger;   ### SRX1 filters NAT POOL1
  ###           route-filter 100.65.0.0/16 orlonger;    ### SRX2 filters NAT POOL2
  ###           route-filter 100.66.0.0/16 orlonger;    ### SRX3 filters NAT POOL3
            }
            then accept;
        }
        term 3 {
```

```
            then reject;
        }
    }
}
routing-instances {
    VR-1 {
        instance-type virtual-router;
        routing-options {
            static {
                route
              100.64.0.0/16
              discard; ### SRX1 installs NAT POOL1 in table
  ###           route 100.65.0.0/16 discard; ### SRX2 installs NAT POOL2 in table
  ###           route 100.66.0.0/16 discard; ### SRX3 installs NAT POOL3 in table
            }
        }

        protocols {
            bgp {
                group srx-to-mx1_TRUST {
                    type external;
                    export trust_export_policy; ### announces 0.0.0.0/0 to MX trust
                    local-as 64500;
                    bfd-liveness-detection {
                        minimum-interval 300;
                        minimum-receive-interval 300;
                        multiplier 3;
                    }
                    neighbor 10.1.1.1 {
                        peer-as 65536;
                    }
                }
                group srx-to-mx1_UNTRUST {
                    type external;
                    export untrust_export_policy;       ### announces clients and NAT POOLs to
MX untrust
                    local-as 64500;
                    bfd-liveness-detection {
                        minimum-interval 300;
                        minimum-receive-interval 300;
                        multiplier 3;
                    }
                    neighbor 10.1.1.3 {
```

```
                        peer-as 65550;
                    }
                }
            }
        }
        interface ae1.0;                        ### Interface assigned to TRUST zone
        interface ae1.1;                        ### Interface assigned to UNTRUST zone
    }
}
security {
    zones {
        security-zone trust {
            host-inbound-traffic {
                system-services {
                    ping;
                }
                protocols {
                    bfd;
                    bgp;
                }
            }
            interfaces {
                ae1.0;
            }
        }
        security-zone untrust {
            screen untrust-screen;
            host-inbound-traffic {
                system-services {
                    ping;
                }
                protocols {
                    bfd;
                    bgp;
                }
            }
            interfaces {
                ae1.1;
            }
        }
    }
    nat {
        source {
```

```
            pool vsrx1_nat_pool {
                address {
                    100.64.0.0/16;     ### example NAT POOL SRX1 or SRXMNHA pair1
    ###             100.65.0.0/16;      ### example NAT POOL SRX2 or SRXMNHA pair2
    ###             100.66.0.0/16;      ### example NAT POOL SRX3 or SRXMNHA pair3
                }
                address-pooling paired;
            }
            rule-set vsrx1_nat_rule-set {
                from zone trust;
                to zone untrust;
                rule vsrx1_nat_rule1 {
                    match {
                        source-address 192.0.2.0/25;
                        destination-address 0.0.0.0/0;
                        application any;
                    }
                    then {
                        source-nat {
                            pool {
                                vsrx1_nat_pool;
                            }
                        }
                    }
                }
            }
        }
    }
    policies {
        from-zone trust to-zone untrust {       ### outbound permit security policy
            policy t2u-permit {
                match {
                    source-address any;
                    destination-address any;
                    application any;
                }
                then {
                    permit;
                    log {
                        session-close;
                    }
                }
            }
```

```
        }
        from-zone untrust to-zone trust {       ### inbound deny security policy
            policy u2t-permit {
                match {
                    source-address any;
                    destination-address any;
                    application any;
                }
                then {
                    deny;                       ### change to permit if appropriate
                    log {
                        session-close;
                    }
                }
            }
        }
        default-policy {
            deny-all;
        }
        pre-id-default-policy {
            then {
                log {
                    session-close;
                }
            }
        }
    }
}
```

NOTE: These sample configurations can use IPv6.

While testing these use cases, some outputs for ECMP CHASH shows the following route selections:

```
user@MX> show route table trust-vr.inet.0 0.0.0.0/0 exact active-path extensive
trust-vr.inet.0: 24 destinations, 28 routes (24 active, 0 holddown, 0 hidden)
0.0.0.0/0 (5 entries, 1 announced)
TSI:
KRT in-kernel 0.0.0.0/0 -> {list: 10.1.1.0, 10.1.1.8, 10.1.1.16 Flags source ip load-
balance}                                                              ### Source-ip LB
        *BGP    Preference: 170/-101
```

```
                Next hop type: Router, Next hop index: 0
                Address: 0x9dd701c
                Next-hop reference count: 2, key opaque handle: 0x0, non-key opaque handle: 0x0
                Source: 10.1.1.1
                Next hop: 10.1.1.0 via ae1.0
                Session Id: 0
                Next hop: 10.1.1.8 via ae2.0
                Session Id: 0
                Next hop: 10.1.1.16 via ae3.0, selected
                Session Id: 0
                State: <Active Ext LoadBalConsistentHash>         ### CHASH used
                Local AS: 64500 Peer AS: 65536
                Age: 50:43
                Validation State: unverified
                Task: BGP_64500_65536.10.1.1.0
                Announcement bits (2): 0-KRT 1-BGP_Multi_Path
                AS path: 64500 65550I
                Accepted Multipath
                Localpref: 100
                Router ID: 10.1.1.0
                Thread: junos-main
user@MX> show route table untrust-vr.inet.0 192.0.2.0/25 exact active-path extensive
untrust-vr.inet.0: 24 destinations, 28 routes (24 active, 0 holddown, 0 hidden)
192.0.2.0/25 (5 entries, 1 announced)
TSI:
KRT in-kernel 192.0.2.0/25 -> {list:10.1.1.2, 10.1.1.10, 10.1.1.18 Flags destination ip load-
balance}                                                                  ### Dest-ip LB
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 0
                Address: 0x9dd821c
                Next-hop reference count: 2, key opaque handle: 0x0, non-key opaque handle: 0x0
                Source: 10.1.1.1
                Next hop: 10.1.1.2 via ae1.1
                Session Id: 0
                Next hop: 10.1.1.10 via ae2.1
                Session Id: 0
                Next hop: 10.1.1.18 via ae3.1, selected
                Session Id: 0
                State: <Active Ext LoadBalConsistentHash>         ### CHASH used
                Local AS: 64500 Peer AS: 65550
                Age: 50:44
                Validation State: unverified
                Task: BGP_64500_65550.10.1.1.2
```

```
                Announcement bits (2): 0-KRT 1-BGP_Multi_Path
                AS path: 64500 65536 I
                Accepted Multipath
                Localpref: 100
                Router ID: 10.1.1.2
                Thread: junos-main
```

NOTE: This configuration is also available in CSDS configuration example as it uses the same technology and configuration for the ECMP CHASH. However, some IP addresses or AS may have changed. For more information, see https://www.juniper.net/documentation/us/en/software/connected-security-distributed-services/csds-deploy/topics/example/configure-csds-ecmp-chash-singlemx-standalonesrx-scaledout-nat-statefulfw.html

Configuration Example for TLB

Like ECMP CHASH, the trust-vr/untrust-vr are similar in the TLB use case, with BGP peering SRX Series Firewalls on each side, however different configuration needs to be made for the TLB services, including additional routing-instances and less policy statements.

Source-hash for forward flow and destination-hash for reverse flow is common for all ECMP based or TLB based solutions. Consistent hash is used during any next-hop failures where it helps an existing session on active next-hops to remain undisturbed, while sessions on down next-hops get redistributed over other active next-hops. This CHASH behavior is pre-built in the TLB solution.

Following sample configuration shows general load balancing strategy for anything but TLB:

```
system {                                        ### internal services needed for TLB
    processes {
        sdk-service enable;
    }
}
policy-options {
    policy-statement pfe_lb_hash {
        term ALL-ELSE {
            then {
                load-balance per-packet;
                accept;
            }
```

```
        }
    }
}
routing-options {
    forwarding-table {
        export pfe_lb_hash;
    }
}
```

The following sample configuration of MX Series Router is for specific forward and return traffic:

```
routing-instances {
    TRUST_VR {
        instance-type virtual-router;
        ### BGP Peering with next router toward clients
        ### BGP Peering with each SRX on the TRUST side (similar to ECMP CHASH)
        interface lo0.1;
        interface ae...;
        interface ...;  ### other interfaces to/from the internal router
    }
    UNTRUST_VR {
        instance-type virtual-router;
        ### BGP Peering with next router toward outside
        ### BGP Peering with each SRX on the UNTRUST side (similar to ECMP CHASH)
        interface lo0.2;
        interface ae...;
        interface ...;  ### other interfaces to/from the external router
    }
    srx_mnha_group_tlb-trust_fi {              ### additional forwarding instance for trust
redirection
        instance-type forwarding;
    }
    srx_mnha_group_tlb-untrust_fi {            ### additional forwarding instance for untrust
redirection
        instance-type forwarding;
    }
}
```

The following sample configuration shows that how traffic is redirected to TLB instance using filter-based forwarding (associated with routing-instance srx_mnha_group_tlb-fi):

```
firewall {
    family inet {
        filter MX_TLB_LB_TRUST {              ### The FBF to redirect traffics to TLB
            term NAPT44_tlb_traffic {
                from {
                    source-address {
                        192.0.2.0/25;
                    }
                }
                then {
                    count SFW44_tlb_forward_traffic;
                    routing-instance srx_mnha_group_tlb-trust_fi;       ### Forwarding instance
used by TLB
                }
            }
            term other_traffic {
                then {
                    count other_traffic;
                    accept;
                }
            }
        }
        filter MX_TLB_LB_UNTRUST {
            term NAPT44_tlb_traffic {
                from {
                    destination-address {
                        192.0.2.0/25;
                    }
                }
                then {
                    count SFW44_tlb_return_traffic;
                    routing-instance srx_mnha_group_tlb-untrust_fi; ### Forwarding instance used
by TLB
                }
            }
            term other_traffic {
                then {
                    count other_traffic;
                    accept;
```

```
                }
            }
        }
    }
}
interfaces {
    ### Aggregate and vlan tagging used on AE (not shown here)
    ae1 {
        unit 0 {
            vlan-id 1;
            family inet {
                filter {
                    input MX_TLB_LB_TRUST;     ### Where forward traffic FBF is applied
                }
                address 10.1.1.1/31;
            }
        }
        unit 1 {
            vlan-id 2;
            family inet {
                filter {
                    input MX_TLB_LB_UNTRUST;### Where return traffic FBF is applied
                }
                address 10.1.1.3/31;
           }
        }
    }
}
```

The following sample configuration shows interface loopbacks used by TLB for health checking to the SRXs:

```
interfaces {
    lo0 {
        unit 1 {
            ip-address-owner service-plane; ### not for RE-TLB but used on other MX
            description "TLB Health-Check Source IP Addresses for TRUST VR";
            family inet {
                address 10.0.0.253/32;
            }
        }
        unit 2 {
```

```
            ip-address-owner service-plane;     ### not for RE-TLB but used on other MX
            description "TLB Health-Check Source IP Addresses for UNTRUST VR";
            family inet {
                address 10.0.0.254/32;
            }
        }
    }
}
```

And the following sample configuration is of the TLB service part (for example, with a NAT service, only trust side TLB instance is used as NAT Pools are announced for return traffic):

```
services {
    traffic-load-balance {
        routing-engine-mode;           ### Important for MX304/MX10K to enable TLB
        instance tlb_sfw_trust {       ### TLB instance for trust forward traffics
            interface lo0.0;
            client-vrf TRUST_VR;
            server-vrf TRUST_VR;
            group srx_trust_group {
                real-services [ MNHA_SRX1 MNHA_SRX2 ];  ### selected SRXs in that TLB group
                routing-instance TRUST_VR;
                health-check-interface-subunit 1;
                network-monitoring-profile icmp-profile;
            }
            real-service MNHA_SRX1 {### Loopback address used on MNHA SRX1 pair
                address 10.0.0.1;
            }
            real-service MNHA_SRX2 { ### Loopback address used on MNHA SRX2 pair
                address 10.0.0.2;
            }
            ...
            virtual-service srx_trust_vs {
                mode direct-server-return;
                address 0.0.0.0;
                routing-instance srx_mnha_group_tlb-trust_fi;  ### Using routes from this VR
                group srx_trust_group;     ### and sending them to that TLB group
                load-balance-method {
                    hash {
                        hash-key {
                            source-ip;                         ### using source-ip as hash
                        }
```

```
                }
            }
        }
    }
    instance tlb_sfw_untrust {     ### TLB instance for untrust return traffics
        interface lo0.0;
        client-vrf UNTRUST_VR;
        server-vrf UNTRUST_VR;
        group srx_untrust_group {
            real-services [ UNTRUST_SRX1 UNTRUST_SRX2 ];
            routing-instance UNTRUST_VR;
            health-check-interface-subunit 2;
            network-monitoring-profile icmp-profile;
        }
        real-service UNTRUST_SRX1 { ### Loopback address used on MNHA SRX1 pair
            address 10.0.0.1;
        }
        real-service UNTRUST_SRX2 { ### Loopback address used on MNHA SRX2 pair
            address 10.0.0.2;
        }
        virtual-service srx_untrust_vs {
            mode direct-server-return;
            address 0.0.0.0;
            routing-instance srx_mnha_group_tlb-untrust_fi;  ### Using routes from this VR
            group srx_untrust_group;            ### and sending them to that TLB group
            load-balance-method {
                hash {
                    hash-key {
                        destination-ip;    ### using destination-ip as hash
                    }
                }
            }
        }
    }
}
network-monitoring {               ### monitor via icmp, http, tcp, udp, ssl/tls…
    profile icmp-profile {
        icmp;
        probe-interval 2;
        failure-retries 3;
        recovery-retries 5;
    }
```

```
    }
}
```

The following sample configuration is of SRX1 for SFW:

```
interfaces {
    lo0 {
        unit 1 {
            family inet {
                address 10.0.0.1/32;
            }
        }
    }
}
policy-options {
    policy-statement trust_export_policy {
        term 1 {
            from {
                protocol direct;
                route-filter 10.0.0.1/32 exact;        ### announce only local loopback
                condition srg_sig_route_exist;  ### if MNHA state is true
            }
            then {
                as-path-prepend "64500 64500 64500";    ### Add more route cost
                next-hop self;
                accept;
            }
        }
        term 2 {
            from {
                protocol direct;
                route-filter 10.0.0.1/32 exact;
            }
            then {
                as-path-prepend 64500;          ### Add less route cost
                next-hop self;
                accept;
            }
        }
        term 3 {
            then reject;
        }
```

```
    }
    policy-statement untrust_export_policy {
        term 1 {
            from {
                protocol direct;
                route-filter 10.0.0.1/32 exact;        ### filter on local loopback
                condition srg_sig_route_exist;  ### if MNHA state is true
            }
            then {
                as-path-prepend "64500 64500 64500";    ### Add more route cost
                next-hop self;
                accept;
            }
        }
        term 2 {
            from {
                protocol direct;
                route-filter 10.0.0.1/32 exact;        ### OR
            }
            then {
                as-path-prepend 64500;          ### Add less route cost
                next-hop self;
                accept;
            }
        }
        term 3 {
            then reject;
        }
    }
}
routing-instances {
    VR-1 {
        instance-type virtual-router;
        protocols {
            bgp {
                group srx-to-mx1_TRUST {
                    type external;
                    export trust_export_policy;
                    local-as 64500;
                    bfd-liveness-detection {
                        minimum-interval 300;
                        minimum-receive-interval 300;
                        multiplier 3;
```

```
                    }
                    neighbor 10.1.1.1 {
                        peer-as 65536;
                    }
                }
                group srx-to-mx1_UNTRUST {
                    type external;
                    export untrust_export_policy;
                    local-as 64500;
                    bfd-liveness-detection {
                        minimum-interval 300;
                        minimum-receive-interval 300;
                        multiplier 3;
                    }
                    neighbor 10.1.1.3 {
                        peer-as 65550;
                    }
                }
            }
        }
        interface ae1.0;                ### Interface assigned to TRUST zone
        interface ae1.1;                ### Interface assigned to UNTRUST zone
        interface lo0.1;                ### Interface used for health check from MX TLB
    }
}
security {
    zones {
        security-zone trust {
            host-inbound-traffic {
                system-services {
                    ping;
                }
                protocols {
                    bfd;
                    bgp;
                }
            }
            interfaces {
                ae1.0;
                lo0.1;
            }
        }
        security-zone untrust {
```

```
            screen untrust-screen;
            host-inbound-traffic {
                system-services {
                    ping;
                }
                protocols {
                    bfd;
                    bgp;
                }
            }
            interfaces {
                ae1.1;
            }
        }
    }
}
```

NOTE: These sample configurations can use IPv6.

While running tests, some output for TLB could be seen as the group usage and packets/bytes to each SRX Series Firewalls:

```
user@MX> show services traffic-load-balance statistics instance tlb_sfw_trust
Traffic load balance instance name    :  tlb_sfw_trust
Network monitor RE daemon connection  :  Established
Interface type                        :  Multi services
Route hold timer                      :  180
Active real service count             :  2
Total real service count              :  2
Traffic load balance virtual svc name :  srx_trust_vs
IP address                            :  0.0.0.0
Virtual service mode                  :  Direct Server Return mode
Routing instance name                 :  srx_mnha_group_tlb-trust_fi
Traffic load balance group name       :  srx_trust_group
Health check interface subunit        :  1
Demux Nexthop index                   :  N/A (810)
Nexthop index                         :  814
Up time                               :  1d 11:19
Total packet sent count               :  81689080324
Total byte sent count                 :  54701197361434
```

```
Real service     Address       Sts  Packet Sent  Byte Sent     Packet Recv    Byte Recv
MNHA_SRX1        10.0.0.1       UP   40823617372  27336541775865
MNHA_SRX2        10.0.0.2       UP   40865462888  27364655583009
user@MX> show services traffic-load-balance statistics instance tlb_sfw_untrust
Traffic load balance instance name    :  tlb_sfw_untrust
Network monitor RE daemon connection  :  Established
Interface type                        :  Multi services
Route hold timer                      :  180
Active real service count             :  2
Total real service count              :  2
Traffic load balance virtual svc name :  srx_trust_vs
IP address                            :  0.0.0.0
Virtual service mode                  :  Direct Server Return mode
Routing instance name                 :  srx_mnha_group_tlb-untrust_fi
Traffic load balance group name       :  srx_untrust_group
Health check interface subunit        :  1
Demux Nexthop index                   :  N/A (812)
Nexthop index                         :  815
Up time                               :  1d 11:20
Total packet sent count               :  62220054022
Total byte sent count                 :  46245032487920
Real service     Address       Sts  Packet Sent  Byte Sent     Packet Recv    Byte Recv
UNTRUST_SRX1     10.0.0.1       UP   31082481387  23096450231084
UNTRUST_SRX2     10.0.0.2       UP   31137572635  23148582256836
```

Common Configurations for ECMP CHASH and TLB

Some elements of configuration need to be in place for both load balancing methods. The following sample configurations are for TRUST and UNTRUST VR and the peering with each SRX Series Firewalls. It also shows some other less seen configuration elements.

- Following is some of the common configurations when using dual MX Series Router topology:Both MX Series Router calculate same hash value when both have same number of next hops, however this is added in Junos OS Release 24.2 (hidden before).

```
forwarding-options {
    enhanced-hash-key {
        symmetric;
```

```
    }
}
```

Results Summary and Analysis

IN THIS SECTION

This JVD shows that scale-out can leverage the use of essential functions both on the MX Series Router and the SRX Series Firewalls for their respective target usage:

- MX Series Router is used as a load balancer with different options, ECMP CHASH and TLB.
- SRX Series Firewall is used as a security service with simple integration with the MX Series Router.
- Both physical SRX Series Firewalls and virtual SRX Series Firewalls are used the same way.
- Simple network integration using BGP and BFD helps on convergence time.
- The addition of new service nodes in this architecture can help to scale in many directions (performances, scaling, an so on) by simply adding new service nodes without disturbing the global service.

Performance/Scale

Though the maximum possible performance and scale capability of the system is out of scope of the JVD, the validation demonstrated the scale-out property of the complex and ability to demonstrate

linear performance and scale growth by adding new service elements to the complex. Initial test is done with a single SRX Series Firewall pair to a typical combination of traffic (100Gbps) as a baseline, a second SRX Series Firewall pair is added to the first one to validate the addition of the same capacity that the first pair is handling (see table later showing tested scaling/performance per SRX Series Firewall pair).

In this case, performance and scaling linearity is obvious when adding more SRX Series Firewall pairs, as the MX Series Router is agnostic to the number of sessions. The amount of traffic stays within MX-PFE throughput limits, every new MNHA pair adds a similar amount of performance to the scale-out complex.

To understand how to reach a maximum performance/capacity, calculate it with an example. Add any number of SRX Series Firewalls until the capacity of the router is reached (for example 3.2Tbps of forwarding capacity with redundant REs or 4.8Tbps without REs, which is high) or its maximum port capacity (for example 16 x 100GE links per line card, up to two cards with redundant REs or three-line cards without REs).

On the SRX Series Firewall side, scaling also depends on the traffic type and its ability to analyze content. More content, more work it needs. Taking 200Gbps tested would then reach MX304 with two-line cards at 3.2Tbps / 200Gbps = 16 SRX, or three-line cards at 4.8Tbps / 200Gbps = 24 SRX. The second MX Series Router and other SRX Series Firewall, the second members of each pair as backup to be able to handle a full load in case of large failure.

Counting the number of available ports (without a distribution layer like QFX) would provide MX304 with two-line cards (and 2 RE) * 16 ports = 32 ports, or three-line cards (and 1 RE) * 16 ports = 48 ports. This is within theorical limits as it does not consider the use of aggregate interface (2 ports) per SRX Series Firewalls, which divides that number by 2.

Load Balancing

ECMP Consistent Hashing has shown steady restoration times in milliseconds.

Using TLB on MX Series Router platforms shows that it also works with non-tested MX Series Router here, where TLB uses a control function on the RE (like MX304) or on a service card (for example, MS-MPC for MX240). TLB has been in Junos since Junos OS Release 18.1R1 when BGP acquired multipath function. This connection with BGP results in service providers often using it internally and externally.

TLB scenario is working with restoration timers and shows flexibility in deployment options (like single or dual MX Series Router is used) as well as a better handling of SRX Series Firewalls in the MNHA cluster.

Security Services

The SRX Series Firewalls features leveraged in this JVD focuses on stateful firewall and CGNAT and did not get into higher layer security features. The fact that scale-out architecture can handle standalone and SRX clusters, using an even distribution among multiple SRX Series Firewalls, without disturbing traffic, shows that the SRX Layer 7 security service can easily be added to this usage.

Note that, with ECMP, all SRX Series Firewalls need to be of the same model whereas, with TLB, this can leverage the notion of TLB groups to have groups of usage (for example, some SRX Series Firewalls in a SFW groups and other SRX Series Firewalls in a CGNAT group). The number of groups is around 2,000 per MX Series Router and the number of SRX Series Firewalls member is around 256. These numbers give a large potential for future use.

Carrier Grade NAT

About CGNAT, the logging capability is not specifically mentioned however can be a key factor. And it is to be noticed that a scalable syslog environment can be set on both sides of the MX Series Router and the SRX Series Firewalls, using their capacity to generate logs at the PFE level, logging at fast rate. Some local laws in various countries need to log a lot of security events and, more simply, what IP addresses have participated in specific events. Then the IP address and NAT attribution is important in those logs. Depending on the CGNAT used and the quantity of security policies actively logging, the solution can generate a fair number of logs.

CGNAT can use deterministic NAT to limit the need for logging related to NAT and port attribution (determined by known algorithm). Or another option is to use PBA (Port Block Allocation) where the ports are allocated during periods and then this attribution event is logged (begin, update, and release).

Each security policy on a SRX Series Firewalls can have the action, “log session-init”, “log session-update”, and “log session-close”. Each action generates a log with the real source/destination IP/port and the NATed source/destination IP/port. This is the feature that may generate the most log quantity. Also, for CGNAT, “category nat” needs to be logged in “security log stream” to record PBA “ALLOC” and “RELEASE” messages.

Scale-Out vs Chassis

This Scale-Out solution is considered as an alternative to the monolithic scale-up approach with the chassis based SRX Series Firewalls or security services on MX960/480 with MX-SPC3 service cards. However, nothing prevents such architectures of being used to benefit from both to leverage the

possibility to add new services and the power of those existing platforms. The upcoming small platforms like MX304 and SRX4700 helps to create a smaller footprint architecture.

Management and Automation

On the management front, configuration automation is not covered. However, it is used to help build and test the solution with various use cases and tests. Basically, scripting is used with Junos access using Netconf. Lots of scripting already exists in the field (or Juniper automation places like GitHub) using Ansible, Terraform, Python, PyEZ (Python Easy for Junos), and so on. Some advanced users have already scripted their Junos, mostly in the service provider space, where APIs are important to integrate with their own management framework. Tools can be created to help with management and automation, running either on-box (on the router itself as it is Python capable) or off-box on a Linux server.

Security Director (on-prem or Security Director Cloud) have a major place for delivering common configurations to the security service layer (like security policies, address objects, and NAT pools), and for providing visibility on the security events and logs generated by each SRX Series Firewalls.

Routing

Junos integration with BGP peering between the MX Series Router and the SRX Series Firewall, includes the right BFD timers, allows you to create a perfectly matching environment with all Juniper solutions working seamlessly together. The redundancy of each router and security solution allows you to maintain steady traffic while providing for the addition of new capacities in a simple way. Similar configuration statements for MX Series Routers and SRX Series Firewalls allows a simple and seamless management of this solution.

Recommendations

- Service Redundancy Daemon (SRD) - https://www.juniper.net/documentation/us/en/software/junos/interfaces-adaptive-services/topics/topic-map/service-redundancy-daemon.html
- Equal Cost Multi Path (ECMP) - **https://www.juniper.net/documentation/us/en/software/junos/interfaces-ethernet-switches/sampling-forwarding-monitoring/topics/concept/policy-per-packet-load-balancing-overview.html**

- Load Balancing Using Source or Destination IP Address Only - https://www.juniper.net/documentation/us/en/software/junos/routing-policy/topics/task/load-balancing-using-src-or-dst-ip-only-configuring.html
- ECMP CHASH - Consistent Load Balancing for ECMP Groups - **https://www.juniper.net/documentation/us/en/software/junos/interfaces-ethernet-switches/topics/topic-map/understanding-ecmp-groups.html**
- Traffic Load Balancing (TLB) - **https://www.juniper.net/documentation/us/en/software/junos/interfaces-next-gen-services/interfaces-adaptive-services/topics/concept/tdf-tlb-overview.html**
- Junos Symmetrical Load Balancing - **https://community.juniper.net/blogs/moshiko-nayman/2024/06/19/junos-symmetrical-load-balancing**
- Multi Node High Availability - **https://www.juniper.net/documentation/us/en/software/junos/high-availability/topics/topic-map/mnha-introduction.html**
- Connected Security Distributed Services - **https://www.juniper.net/documentation/us/en/software/connected-security-distributed-services/csds-deploy/topics/concept/csds-overview.html**
- ECMP Consistent Hashing with Stateful Traffic Flow - **https://www.juniper.net/documentation/us/en/software/connected-security-distributed-services/csds-deploy/topics/concept/csds-ecmp-chash-singlemx-standalonesrx-scaledout-statefulfw.html**
- Automation and Communities -

  https://github.com/orgs/Juniper/repositories?type=all

  https://community.juniper.net/home/techpost