Automating Dynamic QoS Management with Juniper Extension Toolkit

Providing a new level of automation and control to O3b Networks

Table of Contents

Executive Summary 3
Introduction 3
Juniper Extension Toolkit at a Glance 3
Requirements and Topology 3
Dynamic QoS Solution Using JET 4
Why JET Solutions 5
Conclusion 5
About Juniper Networks 6

Executive Summary

O3b Networks Ltd.[1] is a global data services provider operating a scalable medium Earth orbit satellite constellation. The company provides high-speed and low latency Internet connectivity and mobile backhaul services to emerging markets and selected industry verticals that include oil field operations and maritime communications.

This white paper describes how O3b Networks uses Juniper Extension Toolkit (JET)[2] APIs to directly program the control plane on Juniper Networks® MX Series 3D Universal Edge Routers and dynamically manipulate quality of service (QoS) to maintain a high-quality user experience.

Introduction

O3b Networks uses Dynamic QoS for bandwidth and QoS management for maritime communication to improve overall service quality during unpredictable satellite link conditions such as those that occur in bad weather, or when two ships interfere with each other's Internet service delivery.

By implementing Dynamic QoS on MX Series routers, O3b is able to maintain a consistent, high-quality user experience and guaranteed SLAs in unpredictable satellite link conditions by dynamically programming the Juniper Networks Junos® operating system control plane at high speed.

Juniper Extension Toolkit at a Glance

The Juniper Extension Toolkit (JET) is an API framework that gives application developers direct access to the management and control planes within the Junos operating system in order to automate tedious, repetitive operational tasks. JET opens Junos OS to levels of programmability never thought possible, providing a simple, flexible, and extensible automation tool. Direct access to the control plane allows operators to manipulate behavior on an MX Series router, resulting in changes to the router's data plane.

With JET, users can employ Python as a language to create an application that invokes high-speed changes to the control plane via JET APIs. The Python application (referred to as the "JET app" in this white paper) can run as a daemon on the Junos OS device or on an external server. The combination of JET APIs and the Python programming language enable O3b Networks to provide a consistent Internet experience to cruise ship passengers at sea.

Requirements and Topology

In the O3b network topology depicted in Figure 1, two cruise ships (Ship A and Ship B) are sharing a single satellite beam. Ship A is experiencing fading due to weather conditions, which degrades throughputs compared to Ship B, which is in a better link condition. To alleviate this problem, the goal is to dynamically detect changes in modems on each ship and take the necessary corrective actions on the router(s) to enforce QoS priorities.

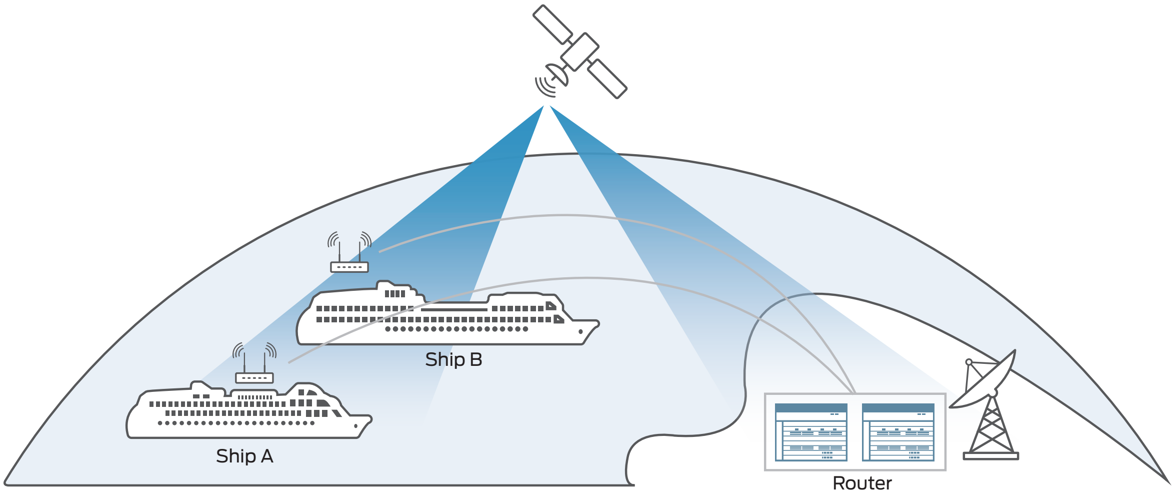


Figure 1: O3b topology

[1] http://www.o3bnetworks.com/

[2] https://www.juniper.net/techpubs/en_US/jet1.0/information-products/pathway-pages/jet-developer-guide.html

At a high level, in order to provide high, stable throughput with consistent Internet access for both cruise ships, O3b Networks requirements include:

- Provide Ship A with double the throughput of Ship B
- Allow ships to burst above their committed information rate (CIR); excess bandwidth should be shared across ships, with Tier 1 bandwidth being prioritized over Tier 2
- Allow classes of traffic within a single ship to have different priorities
- Prevent ships experiencing fading based on weather conditions from a degraded throughput

Dynamic QoS Solution Using JET

To satisfy these requirements, Juniper uses Dynamic QoS as a means of bandwidth and QoS management for this scenario. To achieve a better performance, Juniper exposes the control plane QoS parameters via the JET API and uses Python to build an application called the Dynamic QoS JET app for O3b Networks. The Dynamic QoS JET app performs two basic functions:

- It polls modem data on all ships at predetermined intervals, dynamically computing QoS parameters (i.e., current Modulation and Coding (MODCOD) values based on ship-specific properties.
- It modifies QoS policies on the MX Series router ingress interface for each respective ship (i.e., it performs a configuration and verification process to deploy changes from each polling cycle) in the event the code changes on that ship's modem.

Figure 2 shows the role JET plays in this solution: poll; verify ship stats; compute Committed Information Rate and Peak Information Rate (CIR/PIR); and push policer changes.

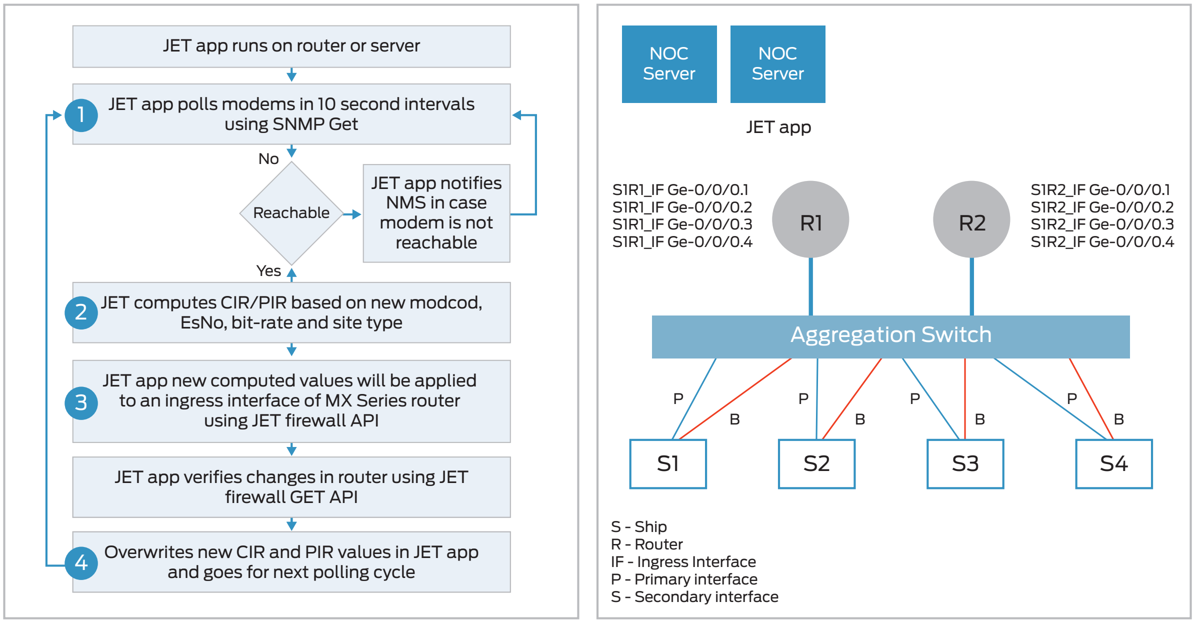
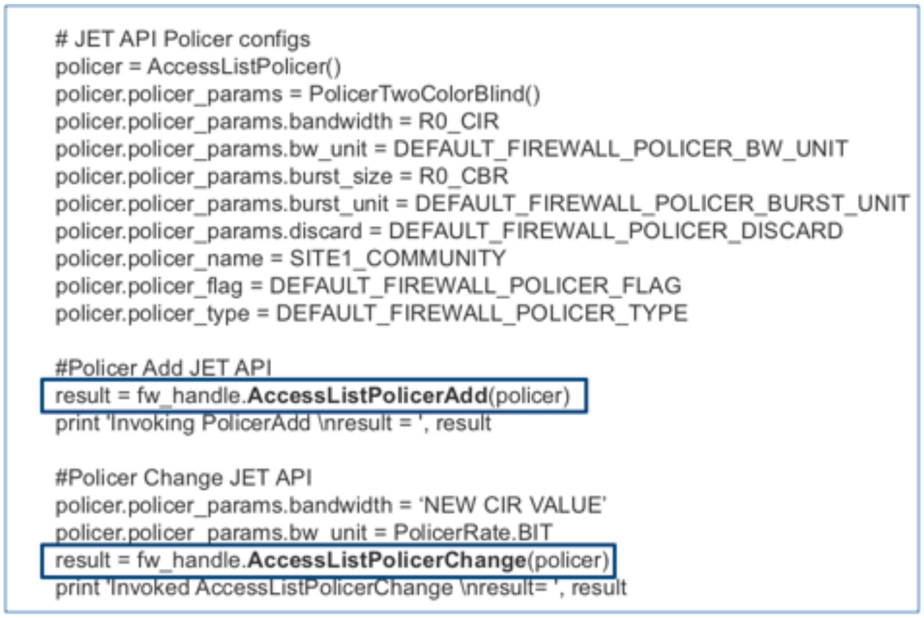

JET app runs on router or server
1
JET app polls modems in 10 second intervals using SNMP Get
No
Reachable
JET app notifies NMS in case modem is not reachable
Yes
2
JET computes CIR/PIR based on new modcod, EsNo, bit-rate and site type
3
JET app new computed values will be applied to an ingress interface of MX Series router using JET firewall API
JET app verifies changes in router using JET firewall GET API
4
Overwrites new CIR and PIR values in JET app and goes for next polling cycle
NOC Server
NOC Server
JET app
S1R1_IF Ge-0/0/0.1
S1R1_IF Ge-0/0/0.2
S1R1_IF Ge-0/0/0.3
S1R1_IF Ge-0/0/0.4
R1
R2
S1R2_IF Ge-0/0/0.1
S1R2_IF Ge-0/0/0.2
S1R2_IF Ge-0/0/0.3
S1R2_IF Ge-0/0/0.4
Aggregation Switch
P
B
P
B
P
B
P
B
S1
S2
S3
S4
S - Ship
R - Router
IF - Ingress Interface
P - Primary interface
S - Secondary interface

Figure 2: Dynamic QoS solution approach with JET

JET APIs can be used to easily push policer configurations and changes, collect policer stat counters, and more. Figure 3 shows a snippet of the JET firewall API.



```
# JET API Policer configs
policer = AccessListPolicer()
policer.policer_params = PolicerTwoColorBlind()
policer.policer_params.bandwidth = R0_CIR
policer.policer_params.bw_unit = DEFAULT_FIREWALL_POLICER_BW_UNIT
policer.policer_params.burst_size = R0_CBR
policer.policer_params.burst_unit = DEFAULT_FIREWALL_POLICER_BURST_UNIT
policer.policer_params.discard = DEFAULT_FIREWALL_POLICER_DISCARD
policer.policer_name = SITE1_COMMUNITY
policer.policer_flag = DEFAULT_FIREWALL_POLICER_FLAG
policer.policer_type = DEFAULT_FIREWALL_POLICER_TYPE

#Policer Add JET API
result = fw_handle.AccessListPolicerAdd(policer)
print 'Invoking PolicerAdd \nresult = ', result

#Policer Change JET API
policer.policer_params.bandwidth = 'NEW CIR VALUE'
policer.policer_params.bw_unit = PolicerRate.BIT
result = fw_handle.AccessListPolicerChange(policer)
print 'Invoked AccessListPolicerChange \nresult= ', result
```

Figure 3: JET firewall API snippet

Why JET Solutions

While other methods were explored as potential solutions to O3b Networks requirements, the JET provided the freedom to explore unique ways to best solve the problem without having to be restructured like other automation tools offered. For example:

- Event scripts can be triggered only for local events (i.e., Juniper routers but not third-party devices).
- Delays of one or more seconds can be imposed on op script invocations to guard against interrupting the collection of events from multiple remote devices.
- Commit times depend on the configuration database; hence, using Network Configuration Protocol (NETCONF) load remote procedure call (RPC) or config commands are not feasible for monitoring modems at every ship and pushing changes to the MX Series routers within 10 seconds.
- Apps based on the features described above cannot be executed from the Dynamic QOS server (i.e., off-box).

JET meets all of these requirements. Highlights include:

- There are multiple ways to detect changes in a modem using JET (i.e., a notification mechanism, polling modems using JET APIs, and the use of third-party tools like pysnmp on-box).
- Dynamic QoS changes can be pushed directly into the control plane in less than a millisecond using JET APIs.
- JET-based applications can be executed on-box as well as off-box, providing greater flexibility.
- The Dynamic QoS JET app can be easily serviced.
- Satellite modems on each ship, MX Series routers, and the Dynamic QoS server are all managed by a single JET app. Without JET, solutions like ship modem stats polling, MODCOD value computing, dynamic QoS configuration activation, and database management for all ships would have to be developed and integrated separately.

Conclusion

O3b Networks offers a tremendous example of how Juniper Extension Toolkit can be used to dynamically program the Junos OS control plane at high speed using a common programming language. The solution provides a better customer experience by guaranteeing Internet SLAs to the cruise lines. Additionally, with the advanced programmability enabled by JET, O3B can now make its network elements aware of those situations and effectively—and dynamically—reprogram QoS settings.

To learn more, please read the “Intro to Juniper’s JET Automation” blog, visit www.juniper.net, or contact your local Juniper partner or sales representative.

About Juniper Networks

Juniper Networks challenges the status quo with products, solutions and services that transform the economics of networking. Our team co-innovates with customers and partners to deliver automated, scalable and secure networks with agility, performance and value. Additional information can be found at Juniper Networks or connect with Juniper on Twitter and Facebook.

Corporate and Sales Headquarters
Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, CA 94089 USA
Phone: 888.JUNIPER (888.586.4737)
or +1.408.745.2000
Fax: +1.408.745.2100
www.juniper.net

APAC and EMEA Headquarters
Juniper Networks International B.V.
Boeing Avenue 240
1119 PZ Schiphol-Rijk
Amsterdam, The Netherlands
Phone: +31.0.207.125.700
Fax: +31.0.207.125.701