WHITE PAPER

ISSU: A PLANNED UPGRADE TOOL

Software and Hardware Implementation Principles

Table of Contents

Table of Figures

Executive Summary

An in-service software upgrade (ISSU) is the ability to migrate from one software version to another in the shortest possible amount of time with minimal risk to network operations. There are multiple ways to accomplish this goal, and vendor ISSU implementations differ substantially in process and results.

Unified ISSU provided by Juniper Networks® JUNOS® Software delivers yet another advancement for reducing the time and risk of planned maintenance events. Unified ISSU enables the complete upgrade from one JUNOS release to another on supported dual Routing Engine platforms with no disruption on the control plane and with no or minimal disruption to traffic. The solution preserves the full integrity of the completeness and quality of the regression testing for each software release and further streamlines upgrades with its automated functions, including compliance checks of the upgrade. The design goals of unified ISSU are an integral part of Juniper Networks commitment to providing continuous systems that deliver high availability and protect network uptime.

This paper is for users who are evaluating ISSU solutions and seeking a deeper understanding of technologies underlying the ISSU process. It provides a general survey of the various approaches to ISSU, noting that an intelligent upgrade mechanism maintains Layer 2/Layer 3 adjacencies and ensures that service-level agreement (SLA) requirements are met. Varying methods for kernel replication present several ISSU options for the control plane. The forwarding plane comes under particular scrutiny, as upgrades to redundant, non-redundant, centralized, and distributed architectures are considered. The latter part of the paper describes the Juniper Networks unified ISSU solution.

Introduction

Resiliency requirements for IP/MPLS networks are steadily increasing with the pressing need to meet business, consumer, and mobile requirements, the high cost of downtime, and the economy's increased reliance on highly available network infrastructures. Convergence, IPTV solutions, and business-critical financial services require accurate and timely transaction delivery, as well as flawless quality of experience. Customers expect services to be available 24/7, and with many of today's "multiple nines" SLA uptime requirements set at six minutes or less per year, networks no longer experience off-peak traffic periods in which to slot scheduled outages for maintenance or upgrades.

Router vendors are addressing these challenges through the introduction of in-service software upgrades, the ability to upgrade a router with no disruption to routing and minimal interruption to packet forwarding. Individual vendors have naturally taken different approaches to ISSU, and the process of choosing among the contrasting techniques can be somewhat perplexing.

Considering the immense variation among today's IP network topologies, equipment, and services, it is not surprising that there is more than one way to perform an ISSU. The right approach is one that addresses the practical problems of today's networks and has the flexibility to meet the needs of tomorrow's networks.

Though the ISSU process is complex and approaches to the problem vary, network administrators have two major goals for the procedure:

- Protocol adjacencies must be maintained.
- SLA requirements must be met.

An in-service upgrade must be transparent to a router's neighbors. A broken adjacency makes it necessary to recalculate routing paths. In the largest networks tens of thousands of protocol adjacencies must be reset and reinitiated, and as many as a million routes removed, reinstalled, and processed in order to establish network-wide forwarding paths.

From the user's point of view, an upgrade mechanism should not affect network topology or interrupt network services. Noticeable packet loss, delay, and jitter can be extraordinarily expensive in terms of SLA penalties and damaged customer confidence.

Approaches to ISSU Design

Several interdependent factors need to be taken into account when designing a tool set for ISSU. Router architecture is a major consideration; to provide a hitless software upgrade with no packet loss, the control plane and the forwarding plane should have physical or virtual redundant components. Because many of today's production routers and networks do not have this capability, an effective ISSU implementation makes allowances for partially redundant systems.

Figure 1 shows the high-level router architecture for typical hardware-based routers. Theoretically, a hitless upgrade would be feasible if all these components were redundant. A non-redundant component would cause some packet loss since it would most likely need to be reset to accommodate a software upgrade.

Upgrade frequency needs to be considered for each component when designing a system for ISSU. If frequent microcode changes are not required for a particular component, the ISSU process could omit that component entirely. However, managing such a process would be difficult given the constant need to track which components should be included in which upgrades. In other cases it might only be necessary to upgrade one component. An example would be a software issue that needed to be fixed within the control plane.

Two basic questions need to be answered before deciding on an appropriate ISSU design:

- What if some components of the forwarding path are not redundant?
- If the new release does not change a given component, should that component be omitted from the ISSU design so the process completes without resetting it?

The answers to these questions depend on the software and hardware architecture of the system and the quality of the operating system development effort. An ISSU design is determined by the quantity and functionality of each component and the interdependence between components.

Control Plane

Recent development work by many router vendors has focused on an effort to provide hitless control plane switchovers, which means keeping the control plane states in sync between the active and standby control planes prior to a switchover. Many consider this capability to be a prerequisite to delivering ISSU. Hitless control plane switchovers are usually implemented using the same version of code on both active and standby control plane components. However, ISSU design additionally requires different software versions running on active and standby control plane components. Supporting different versions requires an instant conversion and synchronization between the old and new states, as well as data structures on different software versions. This requirement further complicates ISSU implementation.

For the purposes of this discussion, view an operating system running on the control plane as being comprised of two main layers: kernel and applications. Once the router has processed updates from all routing protocols and built a routing table, the route calculation result (destination network address, next-hop, and so on) is summarized to the kernel. The kernel is responsible for providing this summary to the hardware, as well as updating control plane applications about any changes on the forwarding plane.

In an effective ISSU design, the active control plane's kernel is replicated on a standby control plane so that control and hardware plane states can be maintained during the upgrade. Design approaches for a hitless control plane switchover include several options:

- An application on the active control plane processes all protocol signaling packets that were sent and received, and then replicates the route calculation results to the standby control plane. The standby maintains its kernel state. This is the most straightforward approach. However, protocol states are not maintained between the active and standby control planes in real time. An inconsistent state can precipitate routing loops during and after the control plane switchover.
- An application on the active control plane resends all protocol signaling packets that were sent and received to the standby control plane. The standby maintains its kernel state. This option is similar, except instead of synchronizing protocol signaling results, the active control plane replicates protocols packets to a standby control plane. This option reduces the delay in synchronization, but it increases the load on the active control plane. If the CPU processing routing protocol updates, the process of resending protocol signaling to the standby control plane might be delayed.

- All protocol signaling packets are mirrored in real time to both the active and standby control plane. Each control plane works independently and serves as a hot standby for the other. This option is the best technical choice, but is difficult to implement. Not all operating system and hardware designs allow this implementation.
- The kernel on the active control plane is replicated to the standby control plane. Graceful restart protocol extensions maintain the protocol state. This option may not be viable for ISSU since graceful restart requires the cooperation of neighboring routers.

Although routing is the most important control plane task, it is not the only one that must be considered when designing an ISSU process. Both active and standby control planes must be aware of configuration and interface state changes after the initial synchronization. A modern, modular operating system allows each process to perform its replication independently, thus simplifying ISSU without compromising the goal of a hitless control plane switchover. For example, for a process that does not require real-time updates for its operation, an active control plane does not have to replicate its states to the standby control plane in real time for the given process.

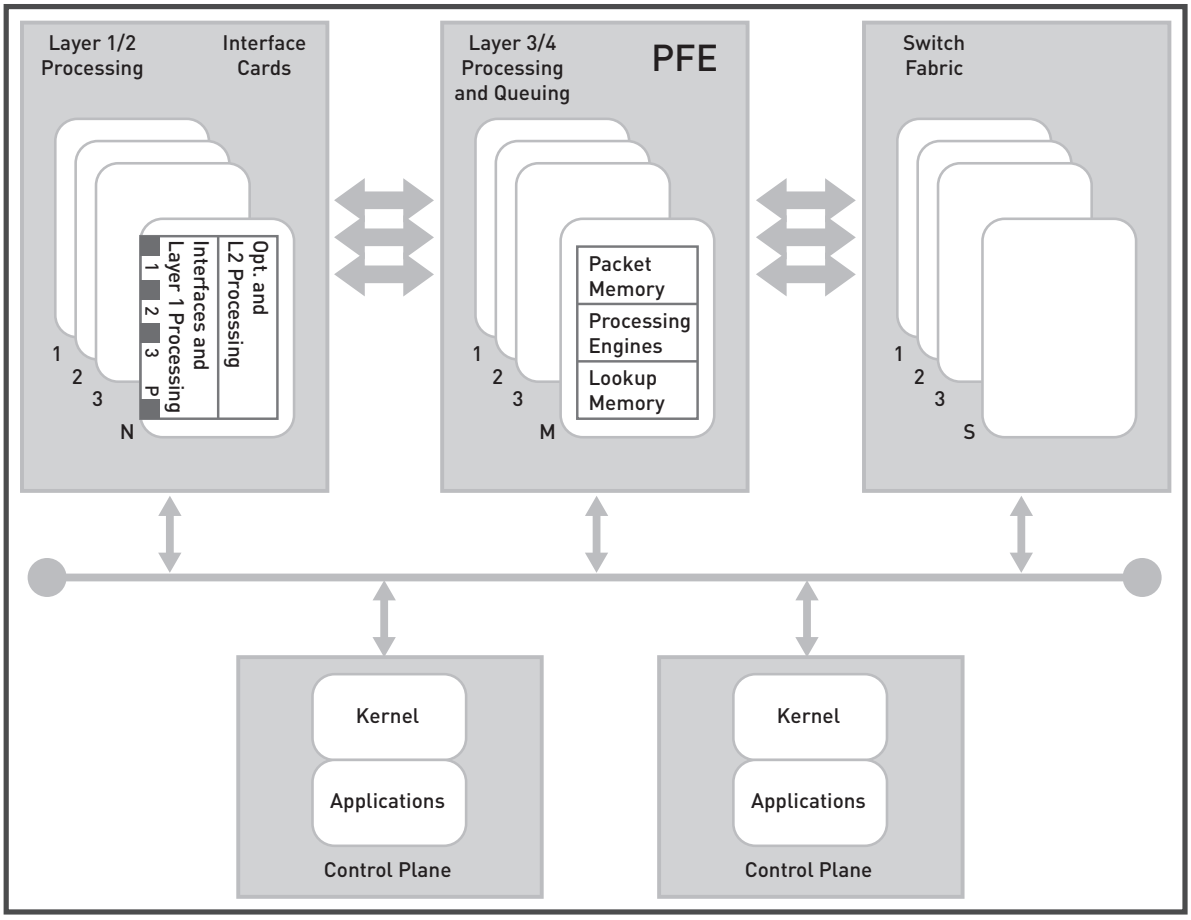

Figure 1: High-level view of a common router architecture

Forwarding Plane

Control plane components in most carrier-class routing nodes are redundant in order to increase system availability. Some of the forwarding plane components may also be redundant, depending on the system architecture. Some have redundant packet forwarding engines, but few have redundant interface cards. Whether or not a particular router component should be redundant depends on its frequency of failure, ease of maintenance, cost, functionality, and availability requirements of the overall system.

Because switch fabric components are usually purpose-built, they do not require frequent upgrades and may be excluded from an ISSU design. The same is true for interface cards, which are generally considered less sophisticated router components.

This section examines the process of upgrading a packet forwarding engine. For certain designs, the principles outlined may be applicable to interface cards and switch fabric upgrades as well.

ISSU Design for Redundant Architectures

It would be ideal if ISSU designers only had to consider redundant hardware on a single hardware platform running a single operating system. In reality, operating systems must support more than one type of hardware; engineers, in turn, have to maintain principles of design that allow an ISSU process to work across all platforms.

Centralized Architectures with Redundant Packet Forwarding Engines

A system with a single active Packet Forwarding Engine (PFE) serving all interface cards can be characterized as a centralized forwarding architecture. An ISSU design is straightforward for centralized systems with redundant PFEs because the backup PFE can be upgraded while traffic continues to flow through the active one. Once the upgrade is performed, the system must be able to switch its interface cards to the upgraded PFE with minimal packet loss. If the system does not have a backup PFE, other creative methods can be used.

Distributed Architectures with Redundant Packet Forwarding Engines

A distributed forwarding architecture is one in which there is a single active PFE that is allocated to one or more interface cards and there is more than one active PFE in the system in any given time. Generally, it is not feasible to design a hardware architecture in which redundant PFEs have a 1:1 binding with the active PFEs. Such a 1:1 backup is not necessary; rather, an N:1 backup is sufficient to meet ISSU goals.

If a system has N:1 backup, the backup PFE is upgraded first. Traffic on the active PFE is then directed to the upgraded PFE, leaving the original backup in standby mode. Since the packet PFEs have to be upgraded sequentially under this design, the total process takes more time than a one-time upgrade. This solution works well as packet drops are kept to a minimum as each PFE is upgraded.

If the system has 1:1 redundant PFEs, the upgrade procedure is not much different than the process of upgrading a centralized architecture. All backup PFEs may be upgraded at once, and traffic then moved to the backup PFEs.

ISSU Design for Non-Redundant Architectures

In today's networks, router forwarding paths always include non-redundant components, and adjustments have to be made to accommodate this common type of architecture.

If a system does not have a backup PFE, a creative ISSU design approach is needed to accomplish the two important initial goals of ISSU: meet SLA requirements and eliminate protocol adjacency flapping. The actual implementation depends on several factors.

A PFE comprises databases, such as the interface database, adjacency database, and forwarding table, as well as processors that use the databases. Figure 2 shows a common PFE design that combines several functions in a single component.

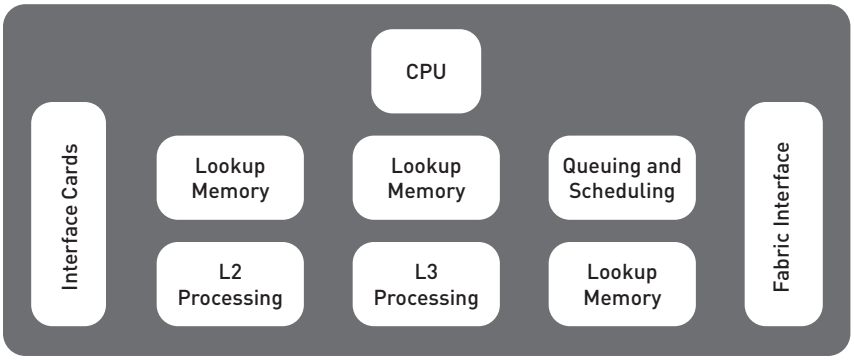

Figure 2: High-level view of a typical PFE design

Each database has to maintain state (counters and values) and configuration throughout the ISSU process. If the new release does not require any changes to the data structure of the databases or any processor configuration changes, it may not be necessary to upgrade some subcomponents. In practice, however, this rarely happens; every major release requires changes to some PFE subcomponents.

To accommodate subcomponent interdependencies, it is good practice to design an ISSU that assumes every new major release will require hardware microcode changes. Since this approach eliminates the possibility of omitting any components from the upgrade, it is stable and predictable. The alternative approach, keeping some components intact during the upgrade, may cause less packet loss if the component is not on the packet forwarding path for some features. Such an implementation must keep track of all these exceptions and can jeopardize system stability. On the other hand, it may be feasible to omit the component from bug-fix upgrades, which only entail software changes or changes to some subcomponents, such as interface drivers.

If a given component does not require an upgrade for a targeted release, the hardware can simply be omitted from the upgrade. In this environment, a software upgrade is not much different than a hitless control plane switchover.

There are basically two ways to accomplish a hardware upgrade if hardware changes are required:

- If a given subcomponent can read the new configuration from memory and execute it without requiring a reset, the subcomponent only requires a warm boot. During a warm boot, the component reads the new configuration and restarts with it. Warm boots can generally be performed quickly and do not cause excessive packet loss.
- If a subcomponent, such as an ASIC, has to be reconfigured for an upgrade and if the new configuration needs to be enforced during the initial startup phase of the component, the component must be reset to change its configuration. This process is called a cold boot. Note that most of the time configuration changes do not require a reset.

Warm boots are the essence of ISSU implementations on non-redundant hardware platforms. Part of the system memory may be allocated to the ISSU, so that the new configuration is included in the target release. Once the switchover occurs, the component reads the new configuration, which is already available in memory.

Cold boots are rare, but do happen. There are two ways to handle a situation in which a given component requires a cold boot to execute the target image:

- Exclude the component from the ISSU and upgrade the other components while the component continues its standard upgrade procedure. This approach is faster than a traditional upgrade, but does not meet the primary ISSU requirement of maintaining protocol adjacencies during the upgrade.
- Assume that the system will not allow an ISSU; the system administrator decides whether or not to proceed with a standard upgrade procedure.

The right implementation choice may be to leave the decision to the system administrators. A good ISSU design should allow the administrator to make an informed decision about any expected outage. If a core subcomponent has to reset, packet forwarding stops during the upgrade procedure, and proceeding with an ISSU may bring little or no benefit. In the rare case that the component is not on the main forwarding path, the administrator may decide to go ahead with the ISSU.

Maintaining Protocol Adjacencies During Upgrades

It is possible to keep Layer 2 and Layer 3 adjacencies up during an upgrade, though how the adjacencies are maintained depends on the details of the implementation. If a system's hardware components are not redundant, the components in service during an upgrade must perform a warm reboot and be reprogrammed. After the warm reboot is completed, the forwarding databases should be updated. Because the warm reboot process is generally much faster than the process of rebuilding the forwarding table, an intelligent ISSU implementation gives priority to the process of fetching essential data from the database. For example, host-bound traffic should continue to be forwarded until the final database synchronization is completed.

Juniper Networks Unified ISSU

The Juniper Networks solution for upgrades is known as unified ISSU. Unified ISSU markedly reduces the time and risk factors commonly associated with software upgrades. Of particular significance are the extensive regression tests that each release undergoes for its ISSU availability, which is only part of Juniper Networks longstanding quality control procedures.

On supported, dual Routing Engine platforms, unified ISSU enables a complete upgrade from one version of JUNOS Software to another without control plane disruption and with minimal disruption of traffic. For example, customers can use unified ISSU today to migrate their Juniper Networks platforms from one JUNOS release to another.

Design Goals

Changes made to the JUNOS Software to support ISSU reflect the fundamental engineering principles of Juniper Networks. These design goals are an integral part of Juniper's overall objective to provide continuous systems that protect network uptime and ensure high availability.

- Mitigate overall risks. An overly complex implementation or a design that requires an exponential verification process to eliminate risks may be more problematic than the traditional method of taking equipment offline during an upgrade.
- Eliminate protocol flapping. Unified ISSU does not impact protocol adjacencies or routing on any of the Juniper Networks platforms.
- Eliminate dependency on other nodes in the network. Unified ISSU relies on nonstop active routing, which prevents protocol adjacency flapping during an upgrade with no requirement for neighbor support.
- Support all hardware types. The unified ISSU design supports both fully redundant and partially redundant upgrade paths without compromising essential engineering principles.
- Remove or minimize the planned software upgrade's affect on SLAs. Unified ISSU keeps protocol adjacencies intact with minimal packet drops on transit traffic.

Software Development Process

JUNOS Software is a single operating system with a single source code base and single implementation of control plane features. Juniper Networks developers and engineers follow a set of development principles to maintain this single train model:

- New features are available only in new releases.
- Maintenance (working) releases are for updates to existing features.
- Juniper Networks does not back-port features to previously released versions.
- Features requested by customers are developed and released in the main line code.

Following these rules means that at all times developers are working with only a single source of code for each release. The result is well understood code with new features and changes that carefully tested for worry-free integration. This single train model is also an advantage in regression testing, which focuses only on the new features rather than the entire train or branches and patches of code.

Regression Testing

The Juniper Networks approach to ISSU ensures that users are running fully supported, fully tested software code. The JUNOS Software testing process includes repeated testing with automated regression scripts, ensuring previously delivered features continue to work as expected in each new release. Developed over many years, these test scripts are critical intellectual property of Juniper Networks. Through extensive testing of each JUNOS release, bugs and other problems are more likely to be found and corrected before customers see the new version.

ISSU regression testing is most reliable when it is only necessary to qualify new features, not multiple versions or feature packages. The JUNOS advantage in this context is its single software release train, which delivers new versions as a superset of features, each passing thorough regression testing with no critical errors.

Complete Operating System Upgrades

A methodical development and testing approach ensures a well understood code base on which to build ISSU operations. Adherence to the single train model allows unified ISSU to replace an entire operating system, not just selected pieces of software or microcode. The solution preserves the full integrity of the completeness and quality of the regression testing for each software release. Upgrade paths are available from any supported release to another.

From an operator's point of view, it is true that control plane code could be singled out for an upgrade patch or bug fix without changing any microcode or resetting any hardware. Such an upgrade could even be hitless in the absence of redundant hardware components. Quality control and regression testing would be difficult under such a system, however. For example, suppose eight software versions are released during a two-year time frame. To support any-to-any releases for all production code, all new software would have to be tested with all eight releases. If each release included 10 critical bug fixes, plus special software packages for each bug fix, each new version would have to be regressed against 8*10=80 software packages to ensure software quality and reliability.

Clearly, no vendor would commit to regression testing on such a scale, despite the fact that without such a commitment, software with code patches applied would be at risk of having major defects and caveats. Uncertainty of this magnitude forces many operations groups to avoid upgrading their software outside of feature or hardware additions, and to continue running outdated code, limiting their options when the network must support new requirements. Additionally, planning around these systems is very difficult when users are waiting for new features.

Juniper Networks has managed all aspects of upgrading in the same way since the company's inception: every JUNOS Software release has been on time since its first distribution in 1998.

Automated Compliance Checks

Unified ISSU streamlines upgrades with its automated operations functions. These functions provide robust messaging to operators, enabling them to correct discrepancies.

Automated compliance checking saves operators significant time and is more accurate than performing manual checks. For instance, unified ISSU initially verifies that all hardware and configured features are supported by the currently installed JUNOS release and are ISSU compatible. It then performs the same verification for the new JUNOS release. Subsequent steps depend on the degree of compatibility:

- System elements supported by ISSU are automatically updated with no or minimal disruption to traffic.
- System elements that are supported by both JUNOS releases but are not ISSU compatible are automatically managed by unified ISSU. Non-compatible elements are taken offline by the ISSU process, updated to the new version while offline, and rebooted to bring them online when ISSU is complete. This procedure does not require operator intervention or support.
- System elements that are not supported by either the new or old version of JUNOS Software, such as new line cards, must be removed from the system before running ISSU, and then reinstalled after the upgrade.

Conclusion

An ISSU enables service providers and enterprises to migrate from one software version to another. The selection of an ISSU design requires a comprehensive approach, one that considers the advantages and disadvantages of various permutations of redundant and non-redundant router architectures.

Juniper Networks unified ISSU design enables service providers and enterprises to smoothly upgrade the entire operating system from one major release to another. For each JUNOS release, the full integrity of the entire code and quality regression testing are achieved. Graceful Routing Engine switchover and nonstop active routing are available on all upgrades. Thus, with Juniper Networks unified ISSU, operators can upgrade software without disrupting Layer 3 adjacencies or routing, Layer 2 keepalives, or link management.

About Juniper Networks

Juniper Networks, Inc. is the leader in high-performance networking. Juniper offers a high-performance network infrastructure that creates a responsive and trusted environment for accelerating the deployment of services and applications over a single network. This fuels high-performance businesses. Additional information can be found at **www.juniper.net**.

Corporate And Sales Headquarters
Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089 USA
Phone: 888.JUNIPER
(888.586.4737)
or 408.745.2000
Fax: 408.745.2100

APAC Headquarters
Juniper Networks (Hong Kong)
26/F, Cityplaza One
1111 King's Road
Taikoo Shing, Hong Kong
Phone: 852.2332.3636
Fax: 852.2574.7803

EMEA Headquarters
Juniper Networks Ireland
Airside Business Park
Swords, County Dublin,
Ireland
Phone: 35.31.8903.600
Fax: 35.31.8903.601

2000280-001-EN Feb 2009