FORWARD NETWORKS

WHY NETWORK VERIFICATION REQUIRES A MATHEMATICAL MODEL

WWW.FORWARDNETWORKS.COM

Introduction

Network verification is a rapidly emerging technology that is a key part of Intent Based Networking (IBN). Verification can help avoid outages, facilitate compliance processes and accelerate change windows. Full-feature verification solutions require an underlying mathematical model of network behavior to analyze and reason about policy objectives and network designs. A mathematical model, as opposed to monitoring or testing live traffic, can perform exhaustive and definitive analysis of network implementations and behavior, including proving network isolation or security rules.

In this paper, we will describe how verification can be used in key IT processes and workflows, why a mathematical model is required and how it works, as well as example use cases from the Forward Enterprise platform. This will also clarify what requirements a mathematical model must meet and how to evaluate alternative products.

Verification: A Key Component of Intent Based Networking

IBN is one of the most interesting and significant trends in IT in recent years. The IBN vision grew out of the need for greater network automation following the partial success of Software Defined Networking (SDN) to simplify cloud deployments and virtual networking. As defined, IBN automates the configuration of networks to align with administrators' high-level intent, as well as the analysis and remediation of network issues.

Today, reasoning in software about the actual behavior of a network and whether or not it has met its design objective is a much more mature technology than recreating the intelligence to design and configure a network to achieve a specific policy requirement on an existing multi-vendor production network. Nearly all successful IBN deployments today are focused on the verification process. The ROI benefits are immediately tangible because many IT processes that verify a network implementation are extremely tedious and can reduce agility or delay network updates significantly.

Verification allows IT teams to automate the analysis of existing network paths end-to-end, based on the collected information (configuration files and state information) from every network device and mathematically analyzing the behavior of all possible traffic flows through each hop. Some examples of end-to-end behavior that IBN can easily verify:

- Are there are least 2 redundant paths from a particular access layer switch to another site through an MPLS Core?
- Are there any single points of failure along an entire network path?
- Have we ensured logical traffic isolation between two tenants or applications?
- Is traffic coming in from the external internet properly restricted to only specific destinations and services?
- Are only specific services running in our Amazon cloud available from various internal sites, systems and users? If so, which ones?

Verification is now fully capable of shifting the network IT model from a reactive approach, to a proactive approach where an automated analysis of the current network implementation can virtually eliminate human errors and misconfigurations. The automated intelligence that IBN offers is also helping to replicate the rare expertise of the critical IT engineers in diagnosing outages, documenting network requirements and verifying fixes.

Verification Automates Key IT Processes

Verifying network configurations manually can be tedious, time intensive and expensive, making it an excellent candidate for IT automation where possible. Since IT teams are now focused heavily on digital transformation and IT automation, the question naturally arises how verification can support these efforts. Three primary areas are commonly addressed by IT organizations:

1. Root cause analysis and accelerating trouble ticket resolution
2. Compliance and audit-related processes
3. Change window validation

Root-Cause Analysis and Remediation of Network Issues

When network issues arise unexpectedly, isolating the root-cause is often a challenge. For example, users and network admins can observe that a certain type of traffic between a source and destination is unable to flow, but the specific device configurations or firewall rules that determine this behavior are hard to identify. Seemingly unrelated changes may have adverse impact to application flows and users in separate parts of the network.

Verification solutions can automate much of the root-cause analysis around anomalous traffic behavior. A detailed analysis of the entire network that can quickly isolate what is preventing a particular flow or behavior can now be completed in a few minutes. IBN deployments are now seeing from 25-50% reduction in time and resources to resolve trouble tickets caused by configuration errors or unexpected changes in the operational state of network devices. In the case of large enterprise networks this can translate to thousands of hours per year.

Compliance and Audit-Related Tasks

Most compliance checks and network audits require verifying key aspects of network behavior, making them prime candidates for process automation through IBN. IBN platforms can verify security policies, such as confirming specific subnets and tenants are isolated, or that all external application access is through HTTPS only. Fault-tolerance and path or device redundancy can also be quickly verified at a glance, with automated checks running continuously, or as frequently as needed.

Verification systems can also automate the search for a wide range of audit-related network health checks, which are difficult to find manually, such as:

- Link speed mismatches
- Maximum Transmission Unit (MTU) size mismatches
- Forwarding loops
- VLAN misconfiguration
- Port channel inconsistencies

Compliance objectives are a natural fit for verification where policy requirements can be specified. Audit-related processes can complete in a fraction of the time. When network snapshots and compliance reports are archived, organizations can easily track compliance results over time and compare to then-current differences in the network implementation. This can give IT organizations a powerful tool to document, track and report on network behavior changes over time.

Change Window Validation and Post-Change Verification

Frequently, the most important times to verify network behavior and capabilities are both before and after a change window. Roughly one-third of all change windows fail because of faulty change procedures, unexpected network conditions, limited test ability or user error. Verifying all network capabilities in both scenarios will immediately expose if there are any adverse or unintended impacts from a set of changes or upgrades.

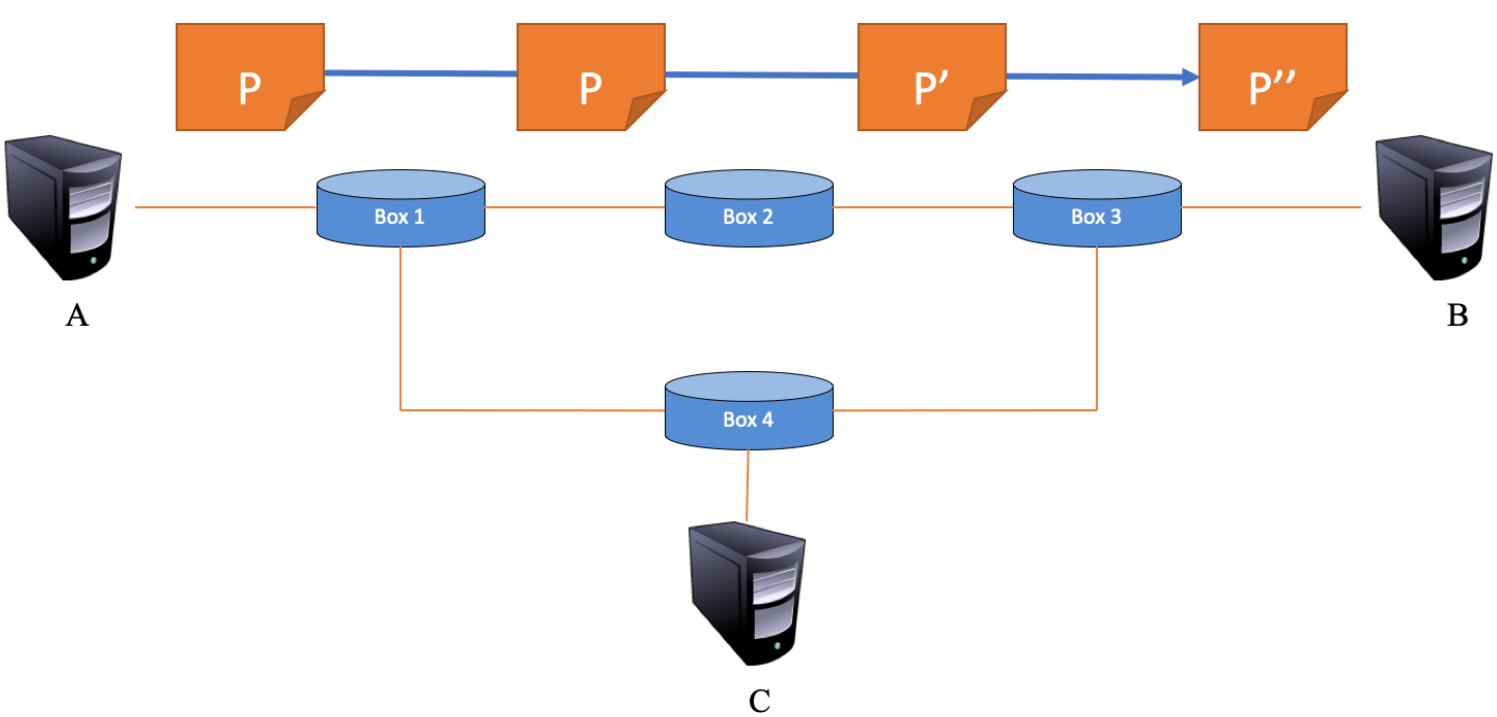
Increasingly, large data center network updates are deployed by automation and orchestration platforms. Automation platforms can repeat configuration tasks hundreds of times, but are rarely fool-proof. IT admins have no ability to perform any kind of verification or oversight at the speed of an orchestration system. Errors can propagate rapidly in the absence of comprehensive verification at the time scale of automation.

What is a Mathematical Model of Network Behavior?

To reason about network behavior as defined above, a system needs a working model of the complete network, incorporating how each device responds to every possible packet. It is referred to as a mathematical or behavioral model of the network because each network device is modeled as a transformation function on a set of potential packets. The transformations are essentially algebraic or logical operations that, when analyzed end-to-end, can verify the complete network design against required policies or behavior.

Let's look at some of the mechanics of these mathematical operations to support network verification. As packets flow from server A to server B, each device in the network can either forward the packet on a particular port, drop the packet, or modify the packet header and forward. In the diagram below, original packet P is transformed to P'' by the time it reaches server B (not every hop may modify a packet header).

Figure 1 – Packets from server A to B are modified at each hop in our network path. Understanding how each device can potentially modify and handle each generic packet is critical to reasoning about possible end-to-end network behavior.



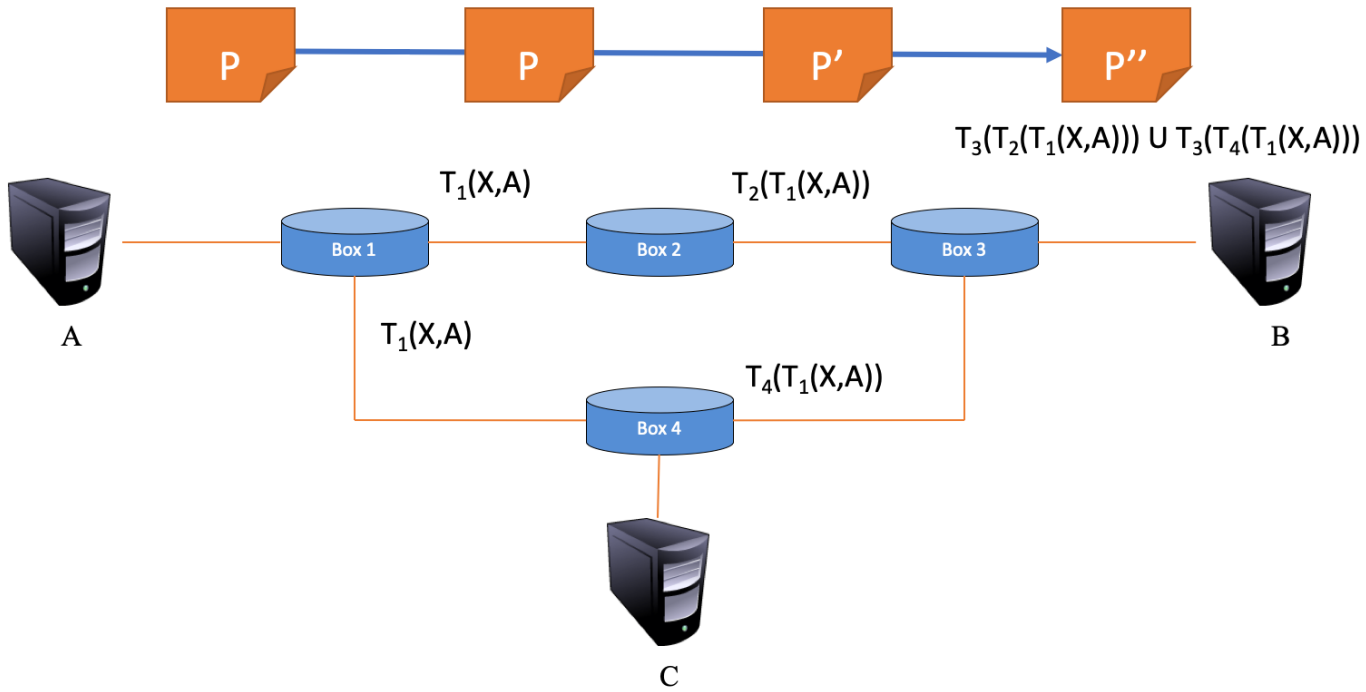


Figure 2 – To determine all the packets that reach server B from server A, we apply successive device transformation functions at each hop from the incoming flow. The results are the union of flows through boxes 2 and 4.

In our mathematical model, we are going to create sets of packets that will have the same behavior at a particular device so that we can ultimately analyze all possible packets in a scalable, manageable way. For our purposes, we are only going to analyze the packet headers and not the data. Generic packet headers are modeled as a binary string, such as 10x1, where "x" can be either a 0 or 1. So, "10x1" (an unrealistically short header used for example), would represent a set of two real packet headers: 1001 and 1011. A more realistic 20-byte header with 100 "x" bits could itself represent over 1030 real packet headers!

Each network device (switch, router, firewall, load balancer) is then modeled as a transformation function on incoming generic packet headers. Transformations usually create multiple sets of transformed packets depending on how many operations and choices the device can make based on the incoming flow. Given an input packet with header h, on port p, the transformation function for a device could be represented as:

$T:(h,p) \rightarrow \{(h_1,p_1), \ldots ,(h_n,p_n)\}$.

The generic packets coming in above result in n different possible results or transformations. Each subset is transformed similarly, with the same set of actions, and then passed to the next hop device in the model.

Every transformation function is a series of rules, in priority order that, when matched to the incoming packet header and port, triggers a series of actions on those packets. Actions may be to drop a range of packet headers, forward a different range to a specific port, or rewrite portions of the header string. For example, for a router with the following route table:

- 172.24.74.x Port 1
- 172.24.96.x Port 2
- 172.67.x.x Port 3

The transfer function, which is breaking up the initial set of incoming packet to three sets of outbound packets, without modifying the header, could be represented by:

$$T:(h,p) \rightarrow \begin{cases} (h,1) & \text{if dst\_ip(h) = 172.24.74.x} \\ (h,2) & \text{if dst\_ip(h) = 172.24.96.x} \\ (h,3) & \text{if dst\_ip(h) = 172.67.x.x} \end{cases}$$

If this device also decremented the time to live (TTL) counter, and rewrote the destination MAC address at this hop, we can modify the resulting headers in our software model of this device and have a resulting transfer function represented as:

$$T:(h,p) \rightarrow \begin{cases} (\text{rw\_mac(dec\_ttl(h), next\_mac)}, 1) & \text{if dst\_ip(h) = 172.24.74.x} \\ (\text{rw\_mac(dec\_ttl(h), next\_mac)}, 2) & \text{if dst\_ip(h) = 172.24.96.x} \\ (\text{rw\_mac(dec\_ttl(h), next\_mac)}, 3) & \text{if dst\_ip(h) = 172.67.x.x} \end{cases}$$

In the above example, dec_ttl and rw_mac are software functions that decrements TTL in the header, and rewrites the MAC address for the next hop. Rule tables for each device are generated from our collection and analysis of the device's configuration files and state tables at the time of the snapshot. See figure 2 for an example of how successive device transformations are applied along an entire path.

Our mathematical model of device transformations is a series of algebraic and logic operations on sets of packets represented by binary header strings (example in figure 3). We are able to then accurately analyze and determine the behavior of all possible packets that could traverse all network paths. Without a mathematical model and underlying algebraic operations, including the accurate modeling of each device based on configuration data, such an exhaustive analysis could not possibly be accomplished.

Scalability is still a crucial design objective. One Forward Networks customer, has a network of over 5,000 devices which translates into over five-octillion (5x1027) viable network paths. Even with such an overwhelming number of paths to analyze, they are able to quickly check whether any of the paths do not conform to stated policies, and to determine the root cause of security or network compliance issues.

The key to a manageable user experience is to perform policy-driven queries that refine the scope of any analysis. More specific queries with path results in the fews tens or even hundreds can be analyzed and manageably presented to users, such as:

- What are all the paths from server A to server B? (see figure 2)
- What are all the destinations from device A (figure 3)
- Are two network zones logically isolated for all protocols but SSH?
- Can any traffic reach a secure zone that bypasses a particular firewall?

Each of these specific queries serves to reduce our analytical space that can be resolved quickly. Despite there being more than six octillion paths in the network, these queries complete in only a few seconds!

Despite there being more than five octillion paths in the network, these queries complete in only a few seconds!

To complete this section, let's look at a specific example query. The generic packet header can be formed from the details of the query, which is used as input to the transformation functions for our current network implementation. How the results are displayed in the Forward Enterprise platform will be shown in later sections.

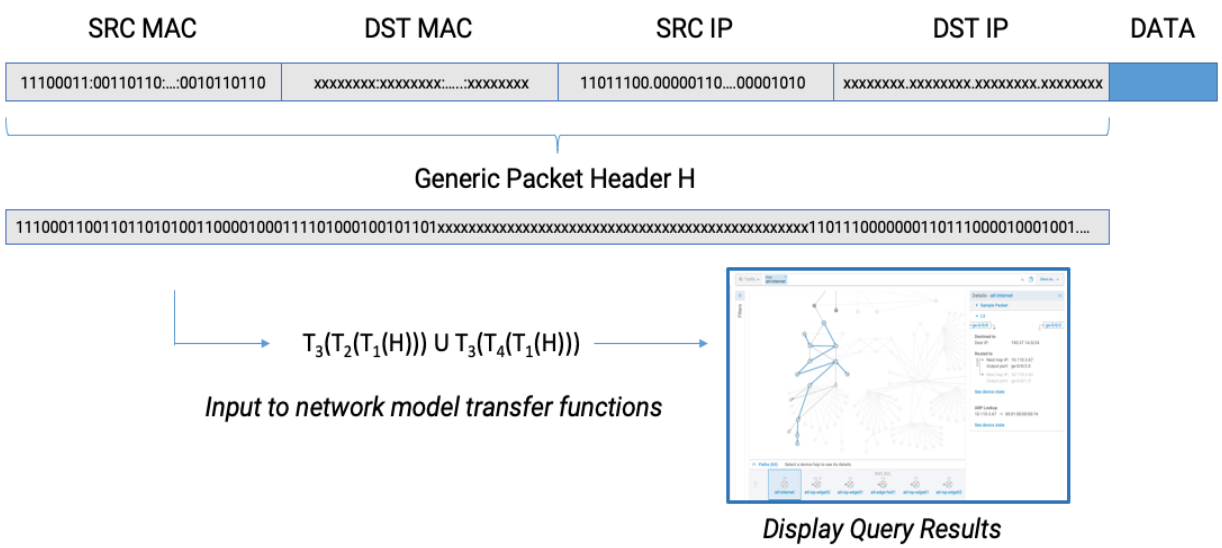


Figure 3 – To determine all the reachable destinations from Device A, we create a generic packet header that fixes the source MAC and IP bits and genericizes the possible destination addresses, and then moves the generic packet header through the network model of device transformations relevant to the source device.

In subsequent sections, we will look at Forward Networks applications and user interfaces, how they leverage this mathematical model to automate analytical processes, and help guide and simplify the user experience.

Powerful Applications Built on the Mathematical Model in Forward Enterprise

The mathematical model forms the deep analytical engine of the Forward Platform. It would not be useful without turnkey applications for network administrators to build their queries that mirrored their actual verification processes and presented the results in an intuitive and actionable fashion. Forward Enterprise has captured a few of the key IT processes and built complete applications for each use case:

- Search (Root-cause analysis and remediation)
- Verify (Validate network designs and proposed updates)
- Predict (Analyze the impact of changes to ACL rules and NAT policies)
- Compare (Analyze changes in configurations and behavior between two points in time)

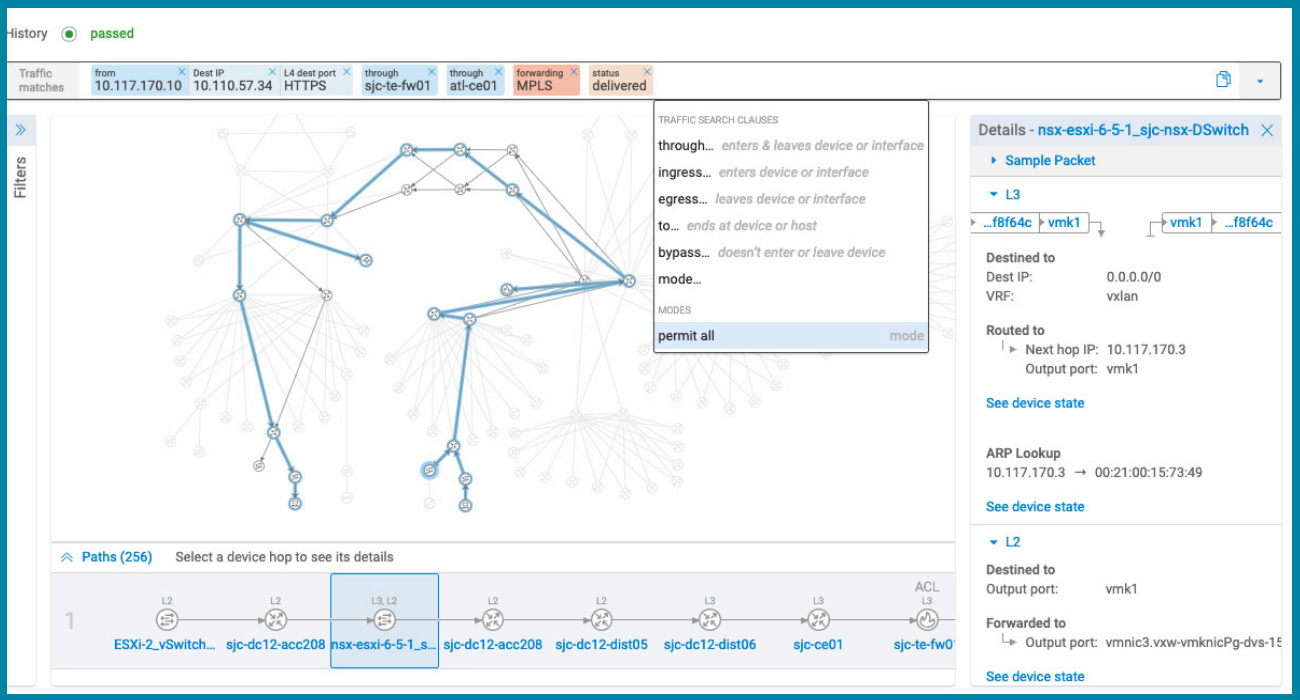
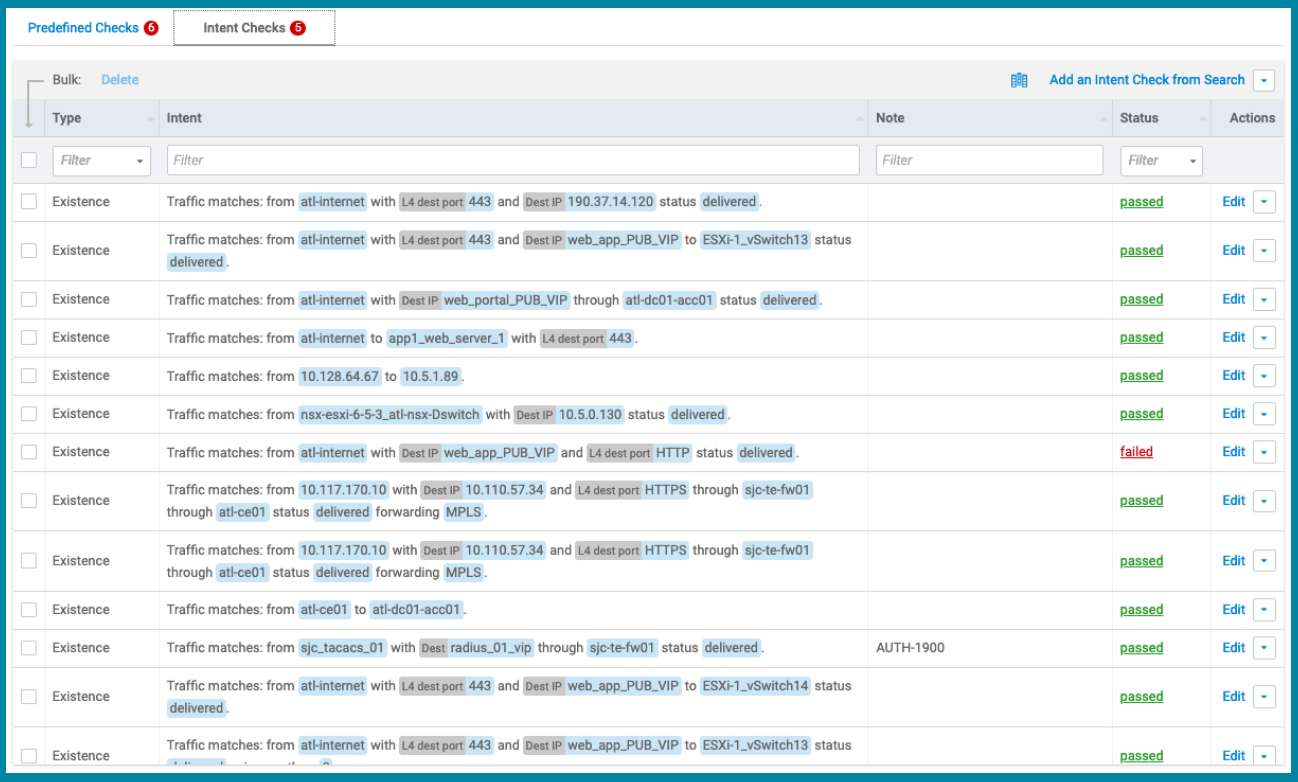


Figure 4 – The search query can be built from modular IP terms and concepts, including source and destination IP, protocols, through devices, delivery status, ports used, etc. A path that supports the search query is displayed within the topology map.

Figure 5 – The Verify screen shows the results of pre-defined policy checks (intent) customized for an enterprise network. Selecting the pass or fail links allows users to quickly drill down to the root cause and potential configuration changes that need to be made.



Search

The Search application in Forward Enterprise allows users to structure queries about the behavior of the current network. For example, is a particular traffic pattern allowed or specifically denied? Search queries can be built with a structured syntax that guides users to include specify policy details and traffic parameters easily, based on well-known concepts in IP networking.

Search queries can start from very broad concepts, such as looking for all devices on a particular VLAN, to very detailed end-to-end policy behaviors as shown in figure 4, with a specific source and destination and through specific devices.

Search is frequently used to isolate and analyze network issues to determine if the network is the root cause, and if so, where the configuration error can be located. It is easy to incrementally refine a search query or expand it to probe down into the network behavior and isolate issues.

The mathematical model is leveraged to translate the traffic query into the appropriate set of generic packet headers to forward through the model. The results of the query, usually a listing of viable paths that meet the search criteria, are displayed on the topology diagram.

Verify

Verify ensures that network intent is actually realized in the production network. Intent is broken down into individual checks, built upon a superset of the syntax used in Search. It ensures the presence or absence of paths in the network that correspond to applications, users, sites, etc.

The Verify dashboard is the result of all prior saved search queries that are re-checked as needed, usually each time a change is made within the network model. Verification checks come in two classes:

- pre-defined, network-independent checks, such as ensuring that IP addresses are unique, there are no forwarding loops, or VLAN definitions are completely consistent
- custom checks for specific networks and policies, such as two subnets should be logically isolated for all traffic but SMTP, or there should always be at least two redundant paths between specific hosts.

For example, if it's a requirement that two edge devices in different data centers are always reachable through multiple redundant paths, that would be saved as a verification check and re-run after every change or update to the network.

Figure 5 shows the results of a number of saved verification checks on a dashboard that can be filtered by pass/fail status or note text.

Predict

Frequently, administrators want to know how a potential change will impact the network prior to pushing to the live network. While Search and Verify are analyzing a snapshot pulled from the network, Predict allows changes to be made, tested and compared within the working software model. Today, Predict supports changes to Access Control Lists (ACL) on switches and routers, firewall rules and Network Address Translation (NAT) services.

Changes to current configuration files are made within the safe sandbox of the Forward Platform and then any search query or verification check can be re-run against the updated model. Comparison of all verification checks can be made side by side against the current network configuration and state information with the proposed changes implemented to fully evaluate before and after change effects.

Figure 6 shows the highlighted lines of configuration code for a set of ACL rules on a particular firewall that we can edit and re-verify within our environment. Without reading through lines of code, the effect of the ACL rules is easily seen in the highlighted column at right.

Compare

Just as Predict can show verification results side by side with current configurations and proposed changes, the Compare feature can compare results and differences between any two snapshots in time. Compare network behavior between today and a month ago prior to issues surfacing to quickly isolate errors. Or show the effects of rolling back changes to any prior network snapshot.

The mathematical model and collected data from each individual device provide immediate documentation and analysis of behaviors at any point in time which can be easily archived for future analysis and comparison. Figures 7 and 8 show a comparison between two snapshots, before and after deploying a new edge firewall. Verification checks are re-rerun and compared side by side (figure 7), as well as showing all the new routes that resulted in the network, or routes that were updated to different hops (figure 8).

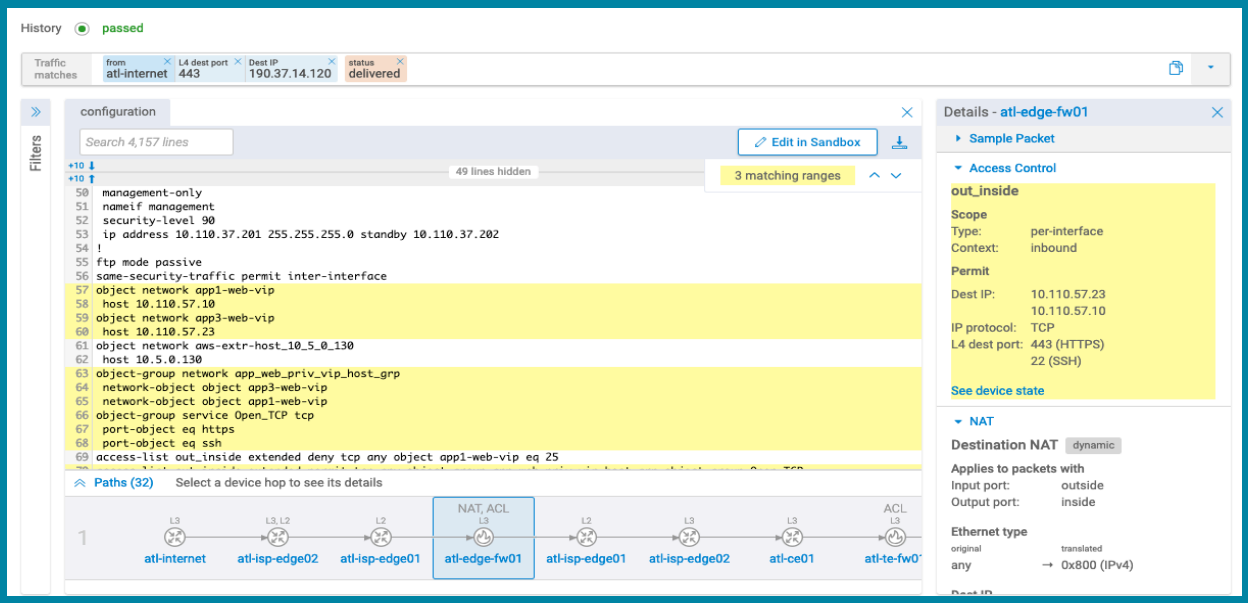


Figure 6 – Make changes to current ACL and NAT configuration files and anticipate changes in network behavior.

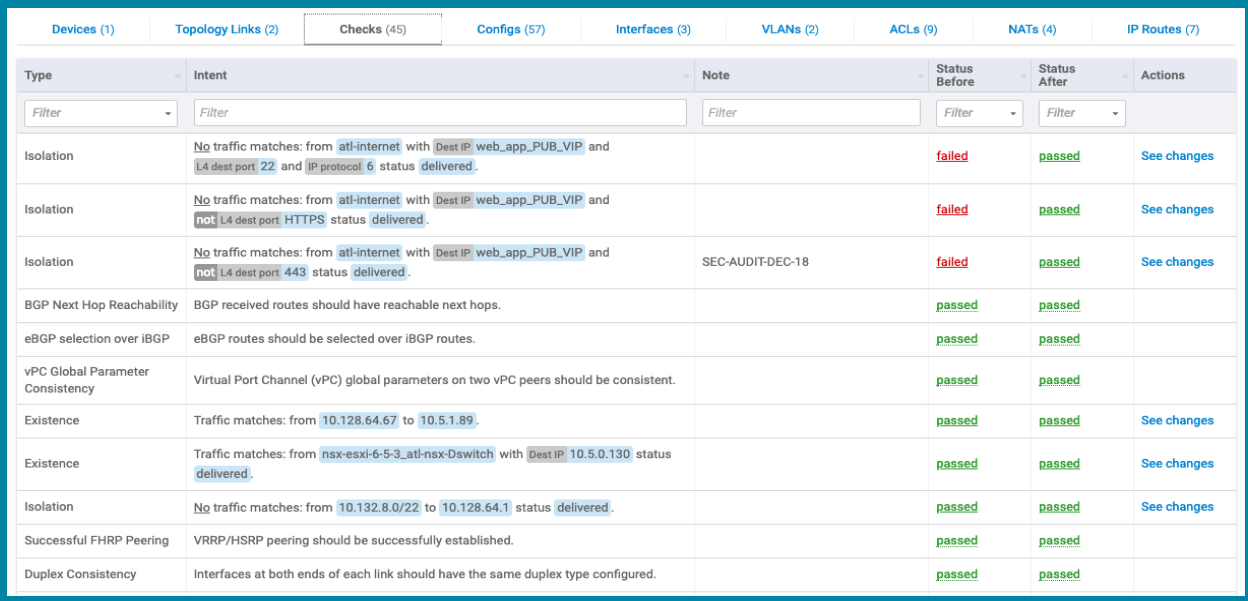


Figure 7 – Compare policy checks side by side between any two network snapshots in time. In this case, key policy requirements are now passing in the “After” snapshot as a result of a change.

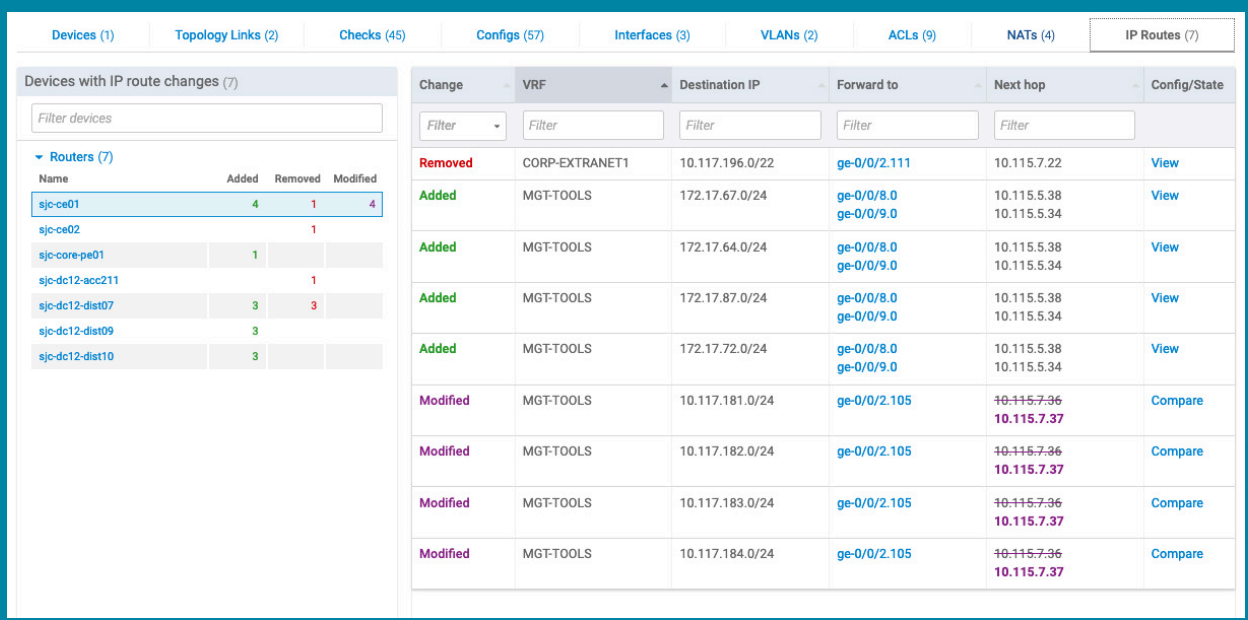


Figure 8 – IP route changes within the network as a result of adding our new device are shown in the above screen capture.

Summary

Intent-based verification is a rapidly emerging technology to ensure that network implementations are aligned with intended policies and requirements. Verification requires an exhaustive analysis of all conceivable packet flows and traffic patterns, which is unrealistic in traditional testing methodologies or evaluating live traffic. A mathematical model that treats every network device as a set of algebraic and logical operations on a large set of packets can now evaluate any and all possible scenarios for a more thorough verification, as well as help isolate the root cause of any behavior issues.

The keys for a successful solution are:

1. Accurate modeling of all network devices, from layers 2 through 4, across all major network vendors and operating systems
2. Scalability in terms of collecting network details from a large number of devices and analyzing or verifying large networks in real-time with a satisfactory user experience
3. Powerful turnkey applications on top of the mathematical model that mirror key IT processes and workflows for remediation, network updates, analysis and verification

Forward Enterprise is the first such highly scalable, multi-vendor network verification solution available today. The sophistication and scale of its mathematical model allows for completely new analytical and verification features compared to existing network management, monitoring or analysis solutions. The automation of key IT processes for remediation, analysis and change verification makes it an ideal solution to complement any network automation project and to return an immediate ROI to large enterprise organizations by reducing manual IT efforts and reducing the risk of network outages.

550 California Ave #200, Palo Alto, CA 94306
www.forwardnetworks.com