USE CASE

Why Network Verification Requires a Mathematical Model

Introduction

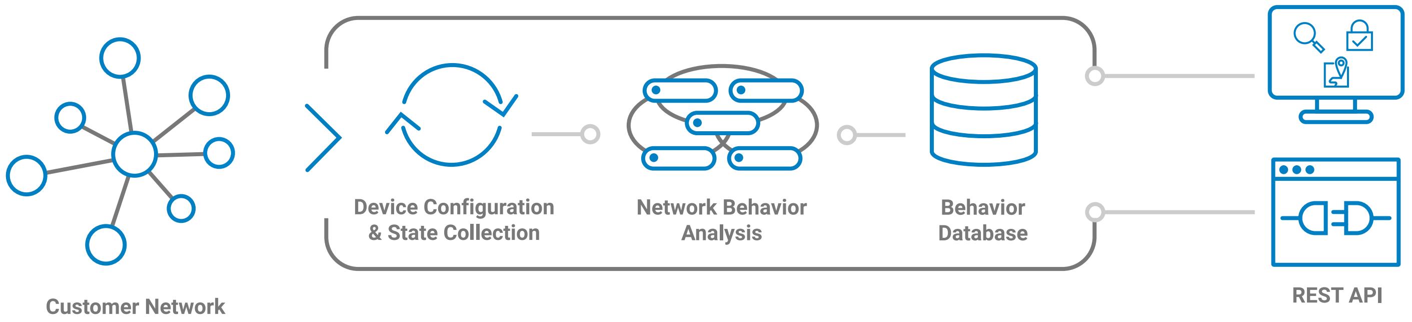
Network verification is a rapidly emerging technology that is a key part of Digital Twin technology. Verification can help avoid outages, verify compliance and accelerate change windows. Full-feature verification solutions require an underlying mathematical model of network behavior to analyze and reason 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: The Cornerstone of Digital Twin Technology

Digital Twin technology is one of the most interesting and significant trends in IT in recent years. By sensing and modeling enterprise networks, a digital twin can replace shadow networks while providing greater confidence that the network won’t fall short. Adoption of digital twin technology is on a steep trajectory with Gartner predicting that by 2025, 25% of enterprises will use digital twins to test part of their network (up from 1% in 2020).

A digital twin is any digital representation of a physical network including emulation, simulation, flow collection, and mapping. When based on a mathematical model, the digital twin can intelligently trace and analyze all possible traffic flows. By collecting a snapshot of device configuration and state then tracing potential traffic via flows, Forward Enterprise builds a “network behavior database” that understands traffic and can answer every question about it – essentially the network becomes searchable like a database.


Customer Network
Device Configuration & State Collection
Network Behavior Analysis
Behavior Database
REST API

Not only can operators query the network to see what is happening, but they can also verify that it's behaving in alignment with the administrators' intent. To truly be of value to enterprises, the technology must scale to support tens of thousands of devices in a single instance. Digital Twins analyze network behavior and provide remediation paths for network issues to reduce MTTR (mean time to resolution).

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. 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 end-to-end behavior queries that a digital twin 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?
- What path does traffic take through the cloud once it egresses the network?
- Are only specific services running in our Amazon cloud available from various internal sites, systems, and users? If so, which ones?

- Is your zone-to-zone connectivity in policy?
- What is the blast radius of a compromised device?
- Are there potential issues in the network, including forwarding loops, maximum transmission unit (MTU) size mismatches, VLAN misconfigurations, or any port channel inconsistencies?
- What are the paths across all relevant transport types? (including Overlay/Underlay visibility, VXLAN, MPLS, Segment Routing, Policy-Based Routing, …)

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 a digital twin 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. Five 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
4. Zone-to-Zone security posture verification including multi-cloud connectivity
5. Blast radius detection
6. Security and behavior verification in hybrid, multi-cloud environments

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. Digital Twin deployments typically reduce the time to resolve trouble tickets caused by configuration errors or unexpected changes in the operational state of network devices from days to seconds. In the case of large enterprise networks, this can translate to thousands of hours per year. One Fortune 50 media and entertainment company reduced 713 hours of work to 38 seconds using their digital twin.

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 using a digital twin. Network digital twins based on a mathematical model 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 then-to-current differences in the network configuration. This can give IT organizations a powerful tool to document, track, and report on network behavior changes over time.

Blast Radius Identification

When a host is compromised, the security team needs to immediately assess the scope of the potential exposure. A digital twin that employs Header Space Analysis to model the network and all possible traffic flows can identify the blast radius of a compromised host in seconds. This allows security professionals to quickly isolate any at-risk devices before the attack spreads.

Zone-to-Zone Reachability Matrix

Zone-to-Zone security matrixes are the heart of an enterprise security policy and the best way to understand if the security posture is in compliance. The addition of a single device or segment can create a complex ripple effect with unintended connectivity changes.

Unfortunately, most organizations don't have a matrix that reflects the actual current state of the network; instead, the zone-to-zone security matrix typically reflects the desired state. A digital twin using HSA with regular collections can present the current state of zone-to-zone interactions in a single, easy to interpret view depicting full connectivity, partial connectivity, or full isolation. This single source of truth becomes assurance that the network is in compliance.

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. Errors can propagate rapidly in the absence of comprehensive verification at the speed and scale of automation.

Cloud Visibility and Verification

Once traffic egresses the on-premises network, it can be extremely difficult to monitor and verify. Each cloud vendor offers proprietary management tools – but they are only designed to work within their environment. The visualization methods and nomenclature are also unique to each cloud vendor, which compounds complexity for those organizations trying to maintain reliability in a hybrid multi-cloud environment.

Using publicly available APIs, Forward Enterprise can collect and normalize data to compute possible traffic flows in Amazon Web Services, Google Cloud Platform, and Microsoft Azure to create a digital twin of the entire estate. Operators can set verification checks to ensure security policy compliance, prevent expensive inter-cloud routing mistakes, and quickly prove network innocence in the event of an incident.

What is a Mathematical Model of Network Behavior?

For a true digital twin that can verify network behavior, discover errors prior to an outage, or compute all possible traffic paths, the digital twin must be based on a mathematical model.

In a mathematical or behavioral model of the network, 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).

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!

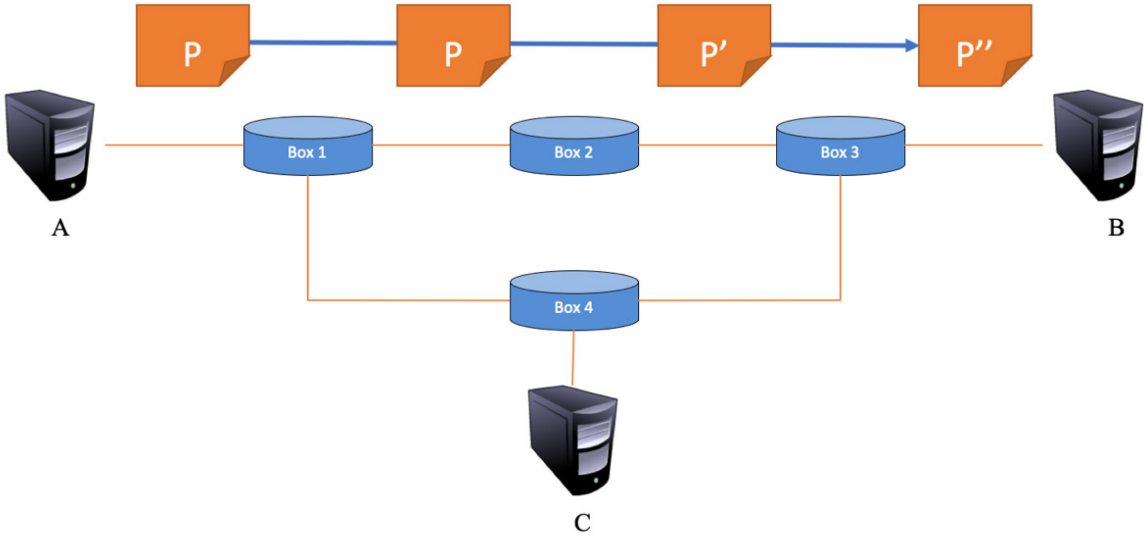


Figure 1 – Packets from server A to B are modified at each hop in our network path. Understanding how each device can potentially modify and handle each generic packet is critical to reasoning about possible end-to-end network behavior.

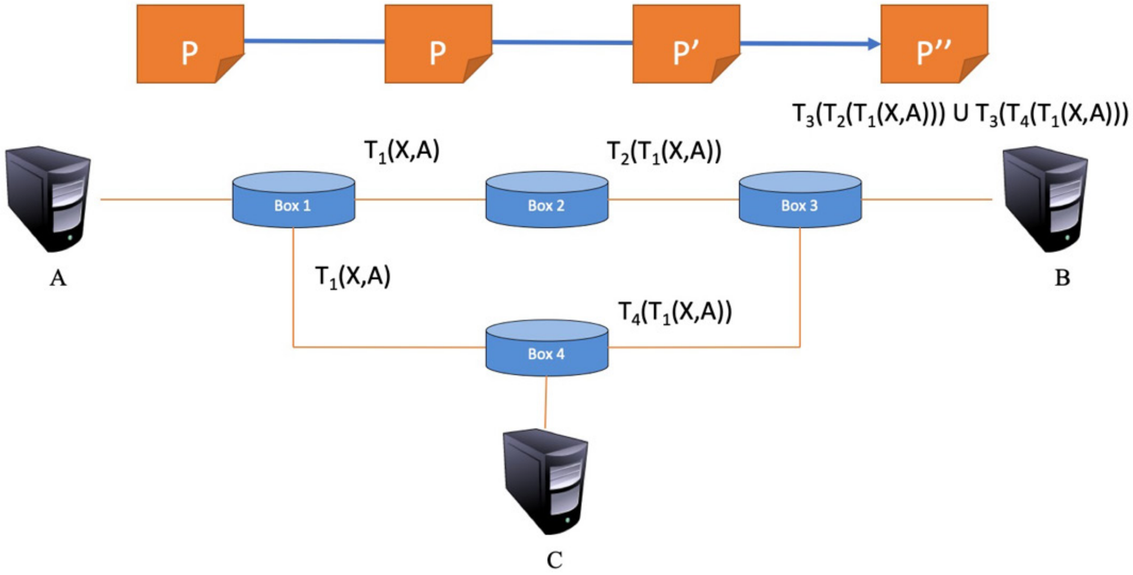


Figure 2 – To determine all the packets that reach server B from server A, we apply successive device transformation functions at each hop from the incoming flow. The results are the union of flows through boxes 2 and 4.

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) ☒ {(h1,p1), … ,(h☒,p☒)}.

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:

(h,1) if dst_ip(h) = 172.24.74.x T:(h,p) ☒ (h,2) if dst_ip(h) = 172.24.96.x
(h,3) if dst_ip(h) = 172.67.x.x

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:

(rw_mac(dec_ttl(h), next_mac), 1) if dst_ip(h) = 172.24.74.x T:(h,p) ☒
(rw_mac(dec_ttl(h), next_mac), 2) if dst_ip(h) = 172.24.96.x
(rw_mac(dec_ttl(h), next_mac), 3) if dst_ip(h) = 172.67.x.x

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.

Forward Networks can support over 50,000 devices in a single instance and compute 10^{19} aggregated 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 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 can be resolved in seconds despite there being more potential paths in the network than there are atoms in the universe.

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.

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.

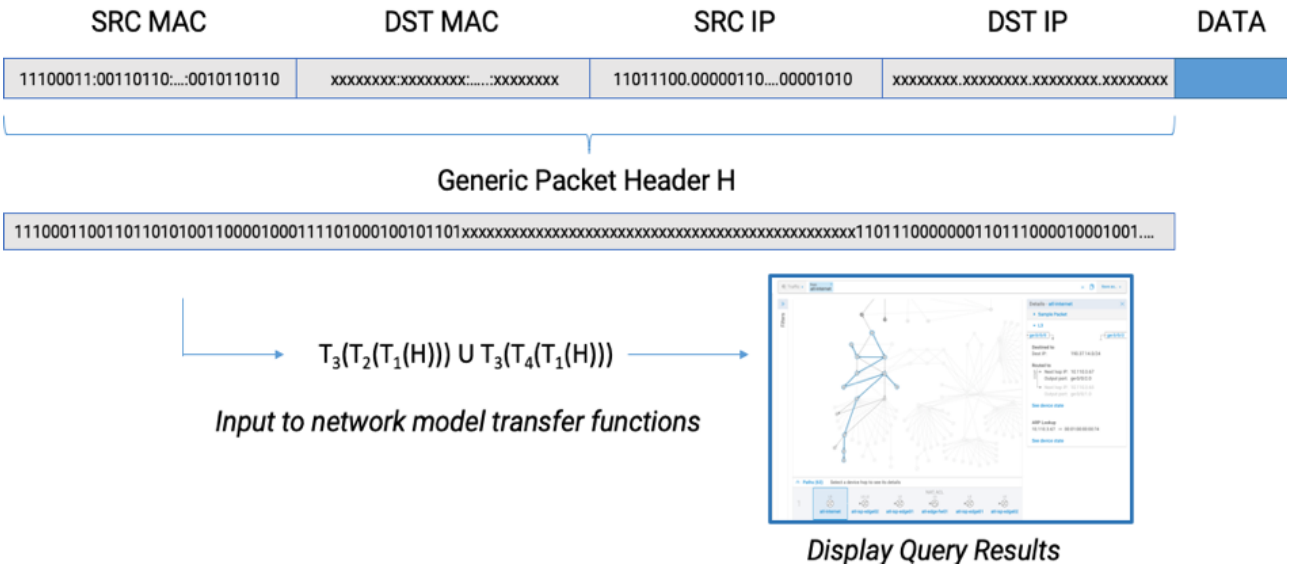
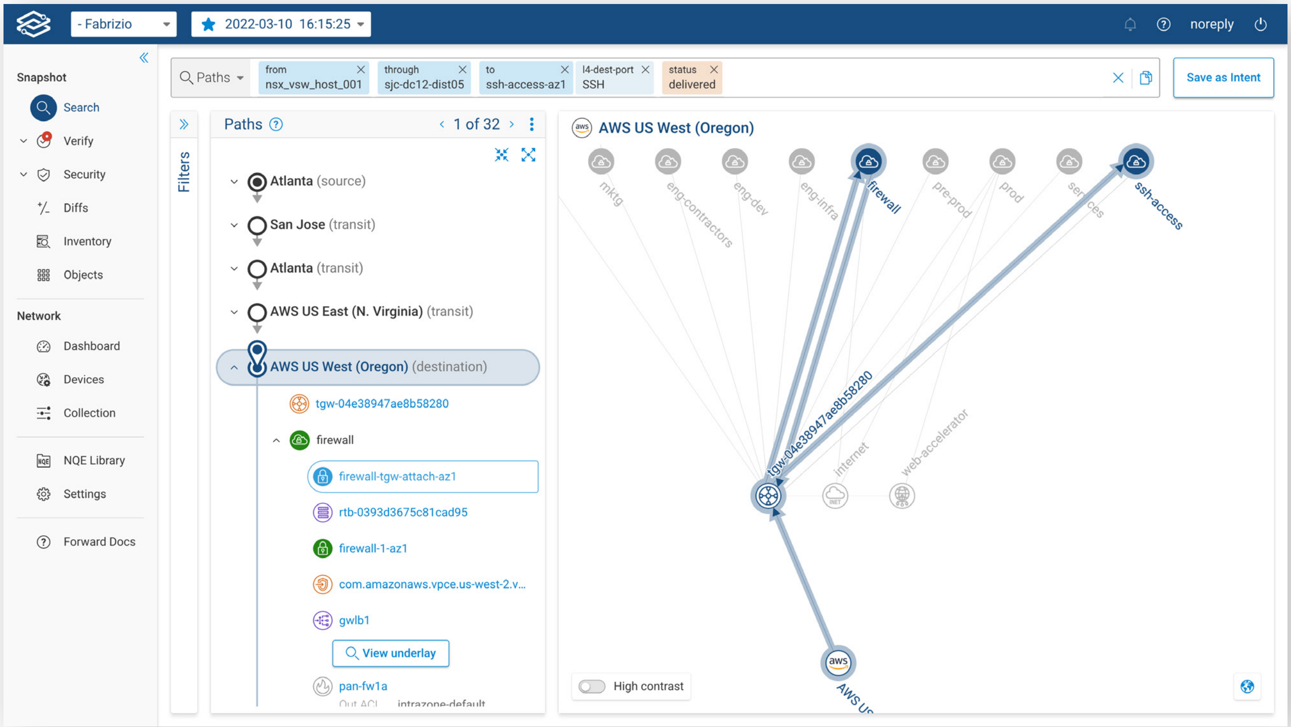

SRC MAC
DST MAC
SRC IP
DST IP
DATA
11100011:00110110:...:0010110110
xxxxxxxx:xxxxxxxx:.....:xxxxxxxx
11011100.00000110....00001010
xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx
Generic Packet Header H
1110001100110110101001100001000111101000100101101xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx11011100000001101110000100010001....
$T_3(T_2(T_1(H))) \cup T_3(T_4(T_1(H)))$
Input to network model transfer functions
Display Query Results

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.

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:

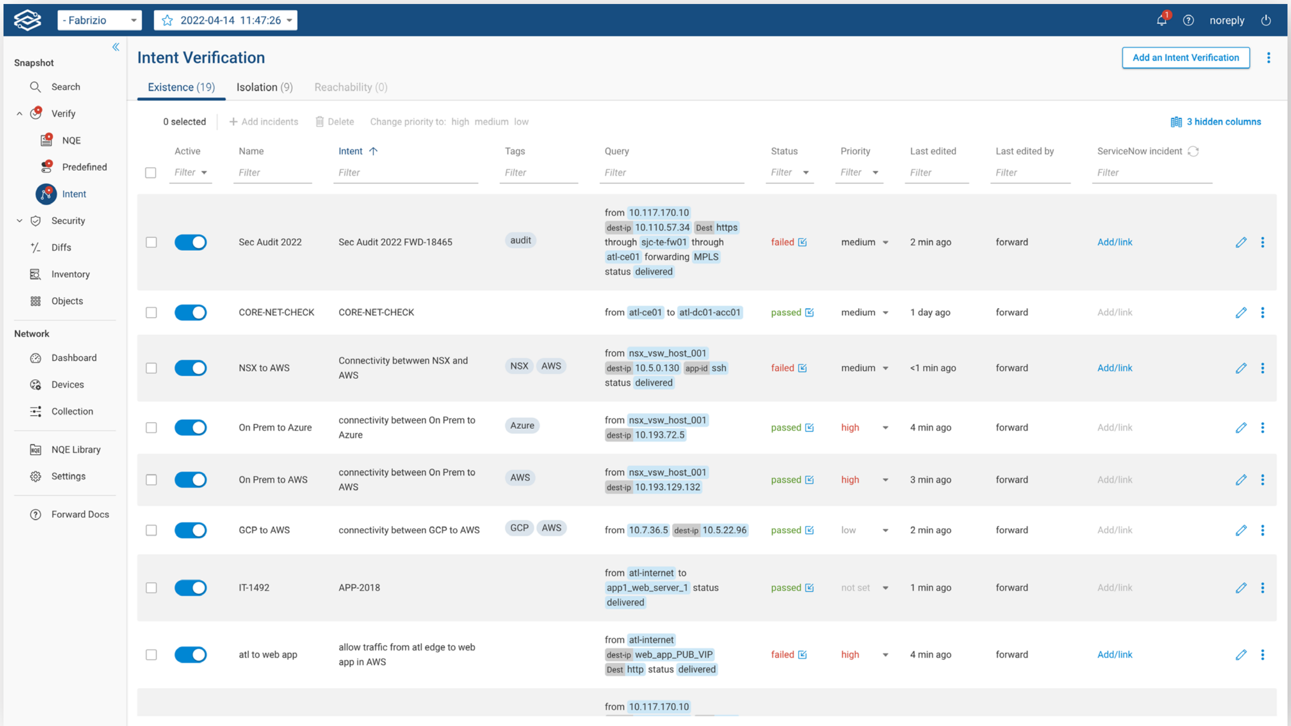
- Search (Perform root-cause analysis and remediation)
- Verify (Prove network behavior is in accordance with intent)
- Predict (Analyze the impact of changes to ACL rules and NAT policies)
- Compare (Analyze changes in configurations and behavior between two points in time)



from nsx_vsw_host_001 | through sjc-dc12-dist05 | to ssh-access-az1 | l4-dest-port SSH | status delivered

- Atlanta (source)
- San Jose (transit)
- Atlanta (transit)
- AWS US East (N. Virginia) (transit)
- AWS US West (Oregon) (destination)
  - tgw-04e38947ae8b58280
  - firewall
    - firewall-tgw-attach-az1
    - rtb-0393d3675c81cad95
    - firewall-1-az1
    - com.amazonaws.vpce.us-west-2.v...
    - gwlb1
    - View underlay
    - pan-fw1a
      Out ACL intrazone-default

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.



Intent Verification

Existence (19) | Isolation (9) | Reachability (0)

0 selected

Active	Name	Intent	Tags	Query	Status	Priority	Last edited	Last edited by	ServiceNow incident
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
●	Sec Audit 2022	Sec Audit 2022 FWD-18465	audit	from 10.117.170.10 dest-ip 10.110.57.34 Dest https through sjc-te-fw01 through atl-ce01 forwarding MPLS status delivered	failed	medium	2 min ago	forward	Add/link
●	CORE-NET-CHECK	CORE-NET-CHECK		from atl-ce01 to atl-dc01-acc01	passed	medium	1 day ago	forward	Add/link
●	NSX to AWS	Connectivity betwwen NSX and AWS	NSX AWS	from nsx_vsw_host_001 dest-ip 10.5.0.130 app-id ssh status delivered	failed	medium	<1 min ago	forward	Add/link
●	On Prem to Azure	connectivity between On Prem to Azure	Azure	from nsx_vsw_host_001 dest-ip 10.193.72.5	passed	high	4 min ago	forward	Add/link
●	On Prem to AWS	connectivity between On Prem to AWS	AWS	from nsx_vsw_host_001 dest-ip 10.193.129.132	passed	high	3 min ago	forward	Add/link
●	GCP to AWS	connectivity between GCP to AWS	GCP AWS	from 10.7.36.5 dest-ip 10.5.22.96	passed	low	2 min ago	forward	Add/link
●	IT-1492	APP-2018		from atl-internet to app1_web_server_1 status delivered	passed	not set	1 min ago	forward	Add/link
●	atl to web app	allow traffic from atl edge to web app in AWS		from atl-internet dest-ip web_app_PUB_VIP Dest http status delivered	failed	high	4 min ago	forward	Add/link
				from 10.117.170.10					

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 their hybrid, multi-cloud environments. For example, is a particular traffic pattern allowed or specifically denied? Search queries can be built with a structured syntax that guides users to easily specify policy details and traffic parameters based on well-known.

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 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.

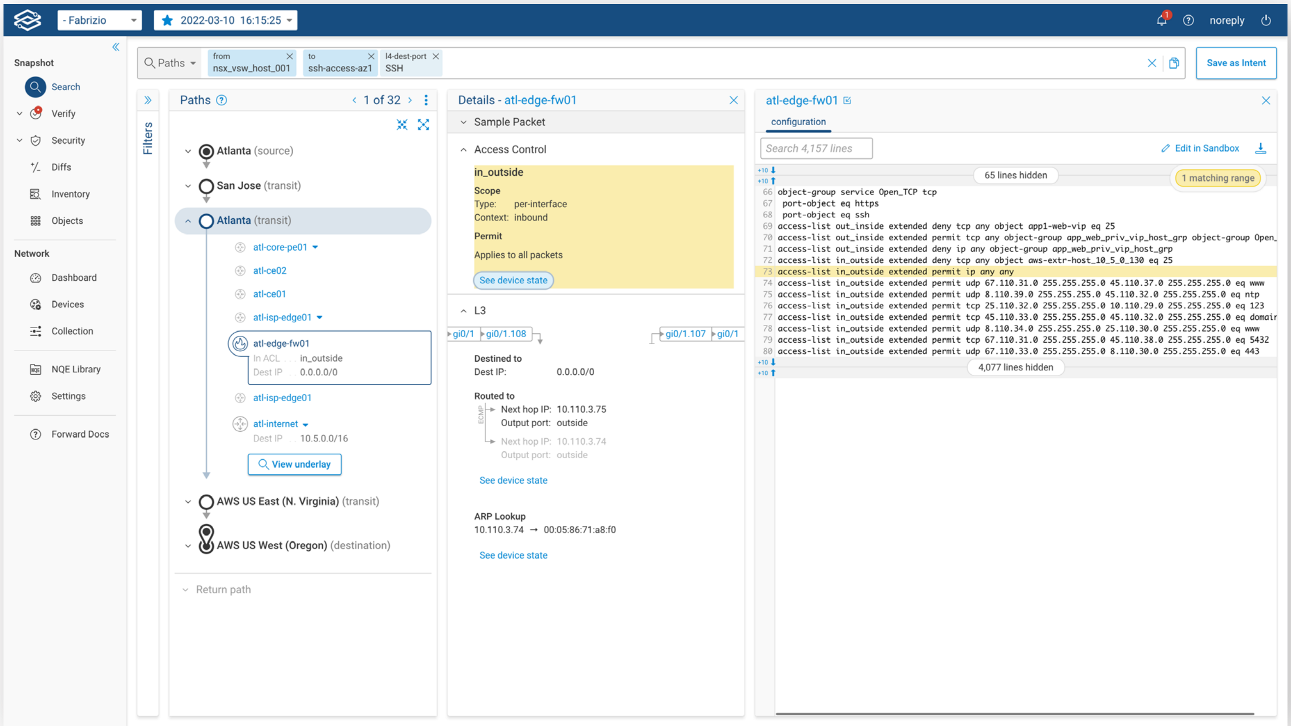
Changes to current configuration files are made within the safe sandbox of the Forward Platform and then any search query or verification check can be re-run against the updated model. Comparison of all verification checks can be made side by side against the current network configuration and state information 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 on the 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).



atl-edge-fw01
In ACL in_outside
Dest IP 0.0.0.0/0

atl-internet
Dest IP 10.5.0.0/16

in_outside

Scope
Type: per-interface
Context: inbound

Permit

Applies to all packets

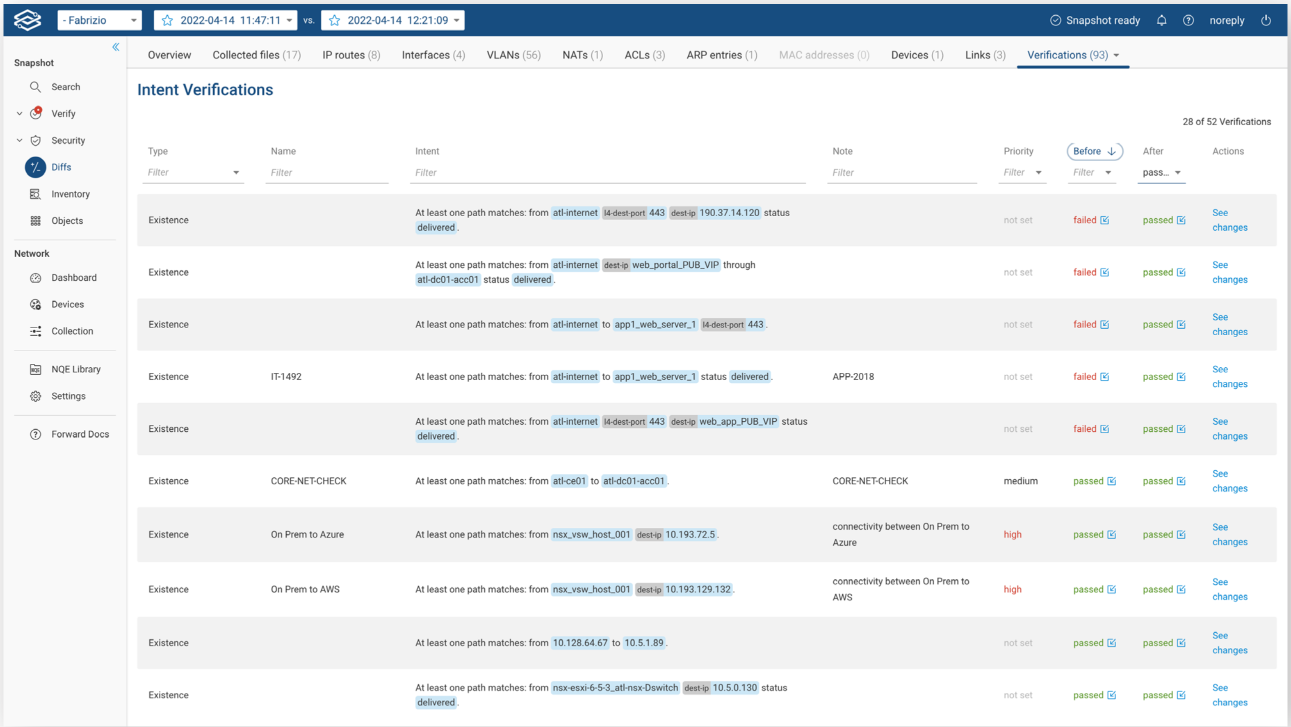
Destined to
Dest IP: 0.0.0.0/0

Routed to
Next hop IP: 10.110.3.75
Output port: outside
Next hop IP: 10.110.3.74
Output port: outside

ARP Lookup
10.110.3.74 → 00:05:86:71:a8:f0

```
object-group service Open_TCP tcp
 port-object eq https
 port-object eq ssh
access-list out_inside extended deny tcp any object app1-web-vip eq 25
access-list out_inside extended permit tcp any object-group app_web_priv_vip_host_grp object-group Open_
access-list out_inside extended deny ip any object-group app_web_priv_vip_host_grp
access-list in_outside extended deny tcp any object aws-extr-host_10_5_0_130 eq 25
access-list in_outside extended permit ip any any
access-list in_outside extended permit udp 67.110.31.0 255.255.255.0 45.110.37.0 255.255.255.0 eq www
access-list in_outside extended permit udp 8.110.39.0 255.255.255.0 45.110.32.0 255.255.255.0 eq ntp
access-list in_outside extended permit tcp 25.110.32.0 255.255.255.0 10.110.29.0 255.255.255.0 eq 123
access-list in_outside extended permit tcp 45.110.33.0 255.255.255.0 45.110.32.0 255.255.255.0 eq domair
access-list in_outside extended permit udp 8.110.34.0 255.255.255.0 25.110.30.0 255.255.255.0 eq www
access-list in_outside extended permit tcp 67.110.31.0 255.255.255.0 45.110.38.0 255.255.255.0 eq 5432
access-list in_outside extended permit udp 67.110.33.0 255.255.255.0 8.110.30.0 255.255.255.0 eq 443
```

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

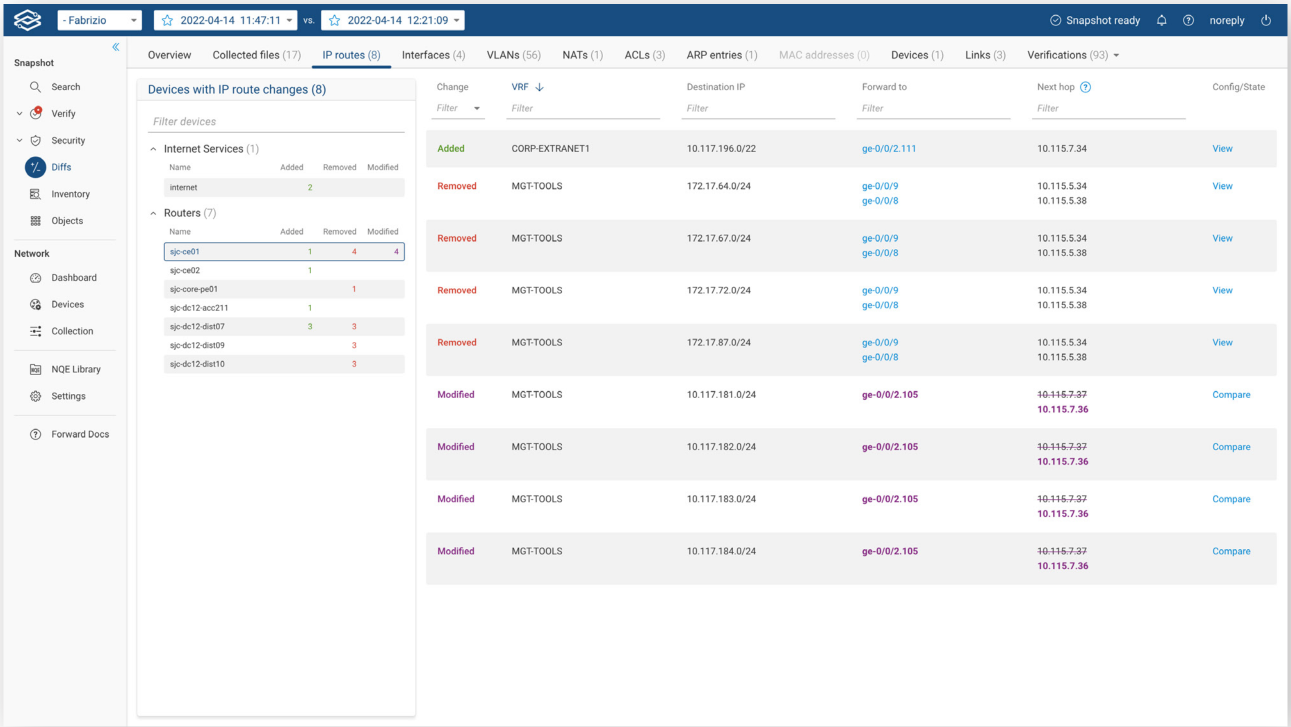


Intent Verifications

28 of 52 Verifications

Type	Name	Intent	Note	Priority	Before ↓	After	Actions
Filter	Filter	Filter	Filter	Filter	Filter	pass...	
Existence		At least one path matches: from atl-internet l4-dest-port 443 dest-ip 190.37.14.120 status delivered.		not set	failed	passed	See changes
Existence		At least one path matches: from atl-internet dest-ip web_portal_PUB_VIP through atl-dc01-acc01 status delivered.		not set	failed	passed	See changes
Existence		At least one path matches: from atl-internet to app1_web_server_1 l4-dest-port 443.		not set	failed	passed	See changes
Existence	IT-1492	At least one path matches: from atl-internet to app1_web_server_1 status delivered.	APP-2018	not set	failed	passed	See changes
Existence		At least one path matches: from atl-internet l4-dest-port 443 dest-ip web_app_PUB_VIP status delivered.		not set	failed	passed	See changes
Existence	CORE-NET-CHECK	At least one path matches: from atl-ce01 to atl-dc01-acc01.	CORE-NET-CHECK	medium	passed	passed	See changes
Existence	On Prem to Azure	At least one path matches: from nsx_vsw_host_001 dest-ip 10.193.72.5.	connectivity between On Prem to Azure	high	passed	passed	See changes
Existence	On Prem to AWS	At least one path matches: from nsx_vsw_host_001 dest-ip 10.193.129.132.	connectivity between On Prem to AWS	high	passed	passed	See changes
Existence		At least one path matches: from 10.128.64.67 to 10.5.1.89.		not set	passed	passed	See changes
Existence		At least one path matches: from nsx-esxi-6-5-3_atl-nsx-Dswitch dest-ip 10.5.0.130 status delivered.		not set	passed	passed	See changes

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.



Devices with IP route changes (8)

Filter devices

Internet Services (1)

Name	Added	Removed	Modified
internet	2		

Routers (7)

Name	Added	Removed	Modified
sjc-ce01	1	4	4
sjc-ce02	1		
sjc-core-pe01		1	
sjc-dc12-acc211	1		
sjc-dc12-dist07	3	3	
sjc-dc12-dist09		3	
sjc-dc12-dist10		3	

Change	VRF ↓	Destination IP	Forward to	Next hop	Config/State
Filter	Filter	Filter	Filter	Filter	
Added	CORP-EXTRANET1	10.117.196.0/22	ge-0/0/2.111	10.115.7.34	View
Removed	MGT-TOOLS	172.17.64.0/24	ge-0/0/9 ge-0/0/8	10.115.5.34 10.115.5.38	View
Removed	MGT-TOOLS	172.17.67.0/24	ge-0/0/9 ge-0/0/8	10.115.5.34 10.115.5.38	View
Removed	MGT-TOOLS	172.17.72.0/24	ge-0/0/9 ge-0/0/8	10.115.5.34 10.115.5.38	View
Removed	MGT-TOOLS	172.17.87.0/24	ge-0/0/9 ge-0/0/8	10.115.5.34 10.115.5.38	View
Modified	MGT-TOOLS	10.117.181.0/24	ge-0/0/2.105	~~10.115.7.37~~ 10.115.7.36	Compare
Modified	MGT-TOOLS	10.117.182.0/24	ge-0/0/2.105	~~10.115.7.37~~ 10.115.7.36	Compare
Modified	MGT-TOOLS	10.117.183.0/24	ge-0/0/2.105	~~10.115.7.37~~ 10.115.7.36	Compare
Modified	MGT-TOOLS	10.117.184.0/24	ge-0/0/2.105	~~10.115.7.37~~ 10.115.7.36	Compare

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, and Layer7 application connectivity 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.

ABOUT FORWARD NETWORKS

Forward Networks' mission is to de-risk and accelerate network operations by increasing efficiency, reducing outages, and verifying network intent. Built on a series of breakthrough algorithms, the Forward Platform provides enhanced network visibility, policy verification, and change modeling for legacy, SDN, or hybrid environments.

Forward Networks is headquartered in Santa Clara, California, and funded by top-tier investors, including Andreessen Horowitz, DFJ, A.Capital, SV Angel, and several luminaries in the networking and systems space.

FORWARD NETWORKS