Converge! NETWORK DIGEST | AvidThink

2019 Next-Generation Network Automation Report

RESEARCH BRIEF

Table of Contents

Enterprise and Service Provider Challenges Today 1

Background and Market Drivers 2

Evolution and Hierarchy of Network Automation Techniques 3

Network Automation Applications and Challenges 5

Recommendations 8

itential®

Intelligent Automation for the Modern Network

Experience the Power of End-to-End Network Automation

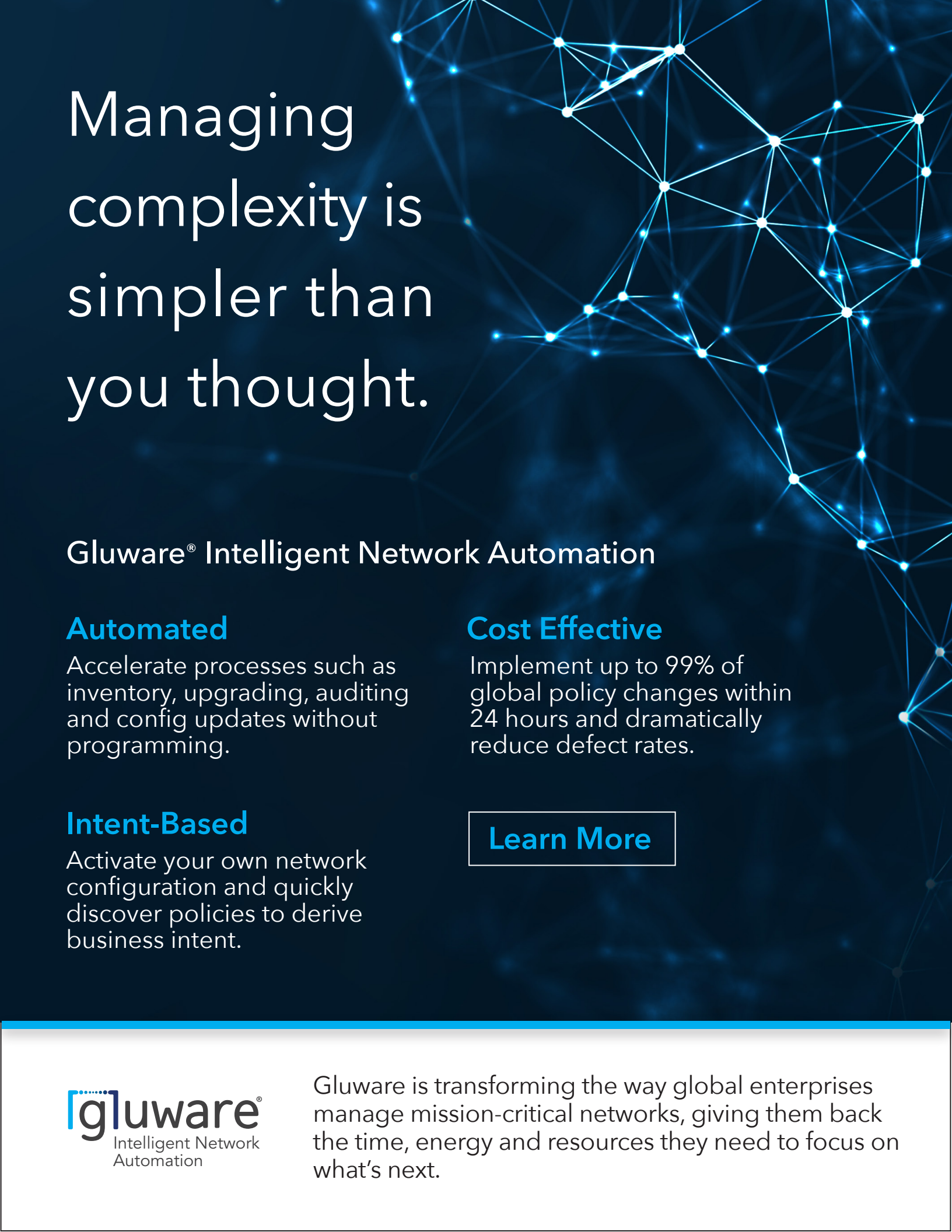
71% improvement in operational efficiency

58% reduction in time to turn up new sites

66% reduction in cycle time for device upgrades

90% increase in software upgrades

itential.com | info@itential.com


Managing complexity is simpler than you thought.
Gluware® Intelligent Network Automation
Automated
Accelerate processes such as inventory, upgrading, auditing and config updates without programming.
Cost Effective
Implement up to 99% of global policy changes within 24 hours and dramatically reduce defect rates.
Intent-Based
Activate your own network configuration and quickly discover policies to derive business intent.
Learn More
gluware®
Intelligent Network Automation
Gluware is transforming the way global enterprises manage mission-critical networks, giving them back the time, energy and resources they need to focus on what's next.

FRINX

THE OPEN SOURCE NETWORK

“Deliver real and sustainable productivity gain by automating processes required to build, operate and grow communication networks.”

https://frinx.io/frinx-machine

frinx.io

RESEARCH BRIEF

2019 Next-Generation Network Automation Report

Enterprise and Service Provider Challenges Today

Whether it's a cause, effect, or a synergistic combination of multiple trends, the explosive growth in cloud services, mobile devices, and broadband network capacity are connected. Keeping up with demands both for more total capacity and higher throughput has been job one at most carriers, ISPs, and hyperscale cloud operators. However, massive increases in network capacity have been accompanied by higher expectations on service providers for faster provisioning of new circuits and network services. The nexus of challenges has forced network operators, whether commercial providers or enterprise IT organizations, to change how they manage and deliver network services.

The most recent **Cisco report on global network usage**, published in 2017, already seems out of date by underestimating the incessant demands from connected devices and network services. Although Cisco predicts total network traffic to double every three years, looking inside the numbers, such as its expectation for average broadband bandwidth or Wi-Fi speed to be just 75 Mbps and 54 Mbps respectively by 2022, makes the predictions look positively quaint.

However, scaling networks hasn't just required component-level improvements in switch silicon and SerDes interfaces, but new, more complicated scale-out network architectures to accommodate many more devices — servers in the data center and mobile devices on wireless networks — while maintaining sufficient redundancy and resilience to cope with the increasing likelihood of equipment failure as the number of required hardware components scales accordingly.

Today's rapid-fire business environment requires organizations, and their suppliers, to **move fast without breaking things**.

Other Stressors: Tight Budgets and Agile Businesses

Unlike other rapidly growing industries, networking providers and internal network teams can't expect generous annual budget increases to fund more staff. Indeed, given the costly requirements of scaling both data center and wireless networks, any extra available money goes to capital equipment. Even here, budgets are tight, leading large network operators to prefer commodity, white box hardware over gold-plated branded products. Lacking more people, network operators must work smarter, not harder.

Even as network operators struggle to keep ahead of capacity needs, they face demanding customers that can't tolerate sclerotic, business-as-usual processes. Instead, today's rapid-fire business environment requires organizations, and their suppliers, to **move fast without breaking things**. Whether you chalk it up to digital transformation, hyperglobalization or **vanishing attention spans**, business today must be more agile in multiple dimensions:

- Faster to develop and introduce new products.
- Quicker to open up new locations and tap new markets.
- Rapidly respond to new competitive and security threats.
- Nimble and creative in exploiting new cloud capabilities, especially in AI and machine learning.

When agile business practices are translated to IT they result in:

- Applications with short release cycles.
- Business data that freely moves across a multicloud, multilocation, low latency network to allow applications to access the best cloud services for the workload.
- Virtual and container workloads that move between physical nodes, both on-premises and in the cloud, in response to automated controllers optimizing capacity utilization and application performance.

All of these factors create the need for networks to be more agile without sacrificing performance or security; requirements that can't be met without network automation.

Background and Market Drivers

Network automation is simply the application of software design and engineering to network configuration, management, and monitoring to improve operational speed, efficiency, repeatability and security. As such, network automation encompasses the core elements of software design including:

- **Resource abstraction** to treat physical elements or concrete parameters as conceptual objects that can be used and manipulated in code.
- **Modeling and architecture** that classifies abstract system components, identifies the relationships between them, and then applies standard design patterns to different types of tasks or processes. These models are often modular to promote reuse across different problem domains.
- **Coding** that uses a structured programming language to turn software models and patterns into specific instructions that are executed per the language's syntax and logic.
- **Program assembly** that bundles one or more software modules into a script or application that implements a particular task or process in response to some external trigger or event.

Interest in all forms of infrastructure automation has increased as a way of coping with the increasing complexity of cloud-scale infrastructure. Indeed, network automation is merely the application of infrastructure-as-code to networking and must work no matter the environment: data center, private cloud, public cloud, campus, WAN or remote office.

Use of network automation arguably accelerated in response to the mainstream commercial failure of layer 2 SDN technologies like OpenFlow — which promised to automate network decision making, traffic management, and configuration via a central control plane — to displace traditional network fabrics in the data center. While SDN has been successful in other realms, notably managing remote connections via SD-WAN, and in optical WAN transports, its use in the data center was hampered by many factors including:

- Inconsistency in vendor implementations.
- Limited hardware resources (e.g., table sizes) in early implementations.
- Limitations in traffic engineering; there's more to traffic management in a modern routing stack than shortest-path algorithms.
- More complexity than expected, which created extensive implementation and management overhead.
- A lack of expertise and the necessary non-siloed IT culture, which exacerbated all of the above factors, leading to many failures during implementation and no appetite to continue to learn, refactor, and invest after hitting the first few roadblocks.

In contrast, the initial goals of network automation were more modest, limited in scope, and easier to achieve, which led to less up-front design and implementation overhead, faster deployments and shorter time to see positive results and achieve

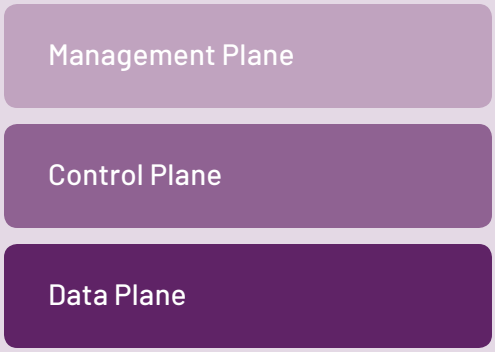
ROI. Unlike SDN, network automation efforts could start small, with simple scripts targeting routine, repetitive processes and build up in complexity by developing higher-level models that worked with a new generation of programmable hardware which exposed all management functions as APIs.

Network Automation looks promising across the board, with market analysts predicting a healthy growth over the next few years. In analyzing results from a recent survey, ACG Research saw an expectation that the market for network automation will grow about 30% annually through 2021, with three-quarters of respondents saying they expect to achieve full or significant network automation in the next five years. And a recent EMA survey of over 250 organizations found that 91% of organizations who had implemented network automation found slight to significant improvement post implementation.

The Relationship between SDN, Network Automation, and Network Orchestration

While we've defined network automation at the beginning of this research brief, we are often asked what the relation between automation and orchestration is. Generally, we view orchestration as a higher-level function that leverages network automation across multiple areas or domains to achieve some goal. For instance, an orchestration system might have a goal of turning on SD-WAN service across multiple different offices, linked via multiple WAN types. It can leverage automation systems for each of those areas, e.g., configuring the network at the branch, provisioning and setting up WAN links, to achieve this goal.

Orchestration and automation are usually focused on provisioning, configuring and managing network equipment, i.e., management plane focused. SDN is usually a different beast altogether, changing the control-plane behavior of a specific network element, or multiple elements, to act differently based on the packet flows that traverse the data plane of a device. In some cases, SDN can be reactive, with real-time user-defined programs that execute as packets come through.

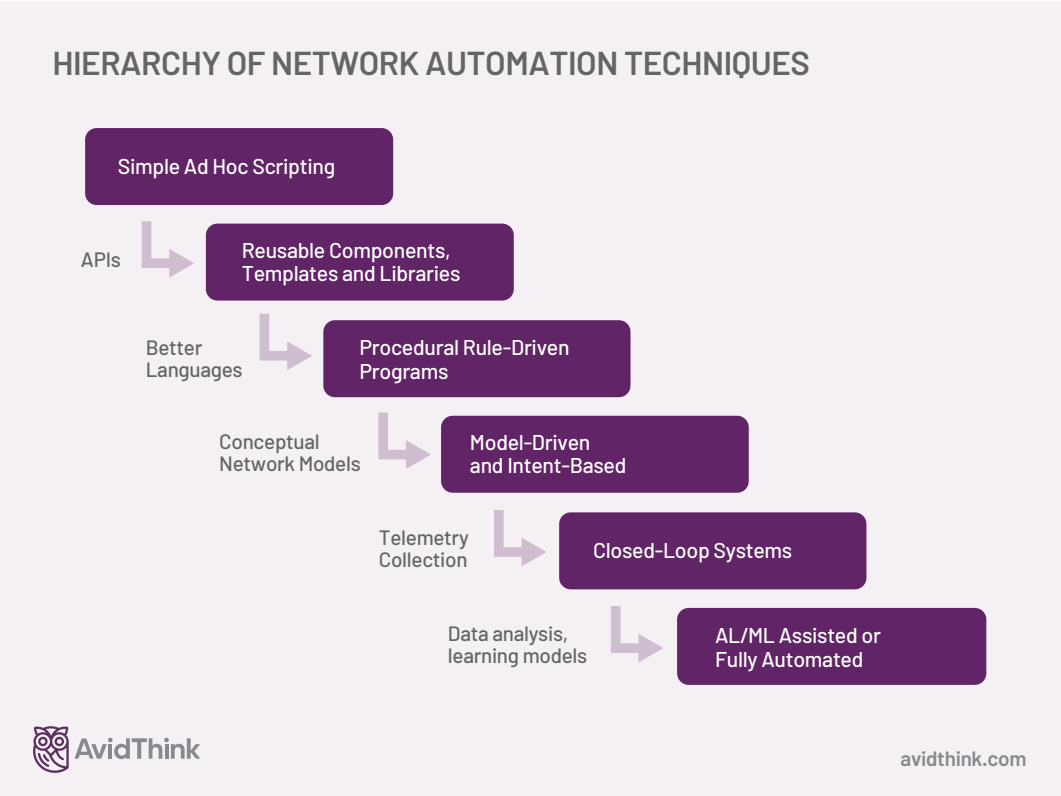


Evolution and Hierarchy of Network Automation Techniques

As mentioned, a key advantage of network automation over SDN and other software-based network architectures is the ability to start small and incrementally work up to more sophisticated models that ultimately, in the case of intent-based and AI-assisted systems, achieve a level of autonomy that can eventually entirely eliminate human activity for particular tasks.

As the following diagram illustrates, network automation starts with basic ad-hoc scripting, which is then augmented with scripting components, libraries, and templates used to build procedural programs. Many network engineers will likely recall writing rudimentary Tcl/Expect scripts against routers and switches, trying to automate a hard-to-automate CLI-based network element, only to have those scripts break on the next firmware release. Over time, these scripts, sometimes written in other languages, would be made into reusable libraries. From there, network engineers evolved into template-driven configuration, separating parameterization from a common set of proven and validated configuration settings that could be

applied, cookie cutter like, across the network. And then logic and procedures were added that could take specific action based around triggers or observed events.



Next up the automation hierarchy are model-driven, intent-based systems in which network architects define network policies and operating conditions, and the system manages the implementation through automated configuration changes and security updates. With intent-based systems, the network operator defines the "what" to be achieved and lets the system figure out the "how" on its own, with little to no human intervention. For example, the intent could be "connect node A in a location one to node B in location two", it may even specify an service-level agreement (SLA) to be met, and the intent will be translated into a series of configuration steps that the system carries out autonomically and key parameters that it will continuously monitor to ensure compliance with. A critical feature of intent-based systems is the ability to adapt to changing workloads, security threats, and capacity demands to keep the network in compliance with the defined network state.

Both intent-based and traditional automation systems benefit from rich telemetry information, which can be transformed into key performance indicators and facilitate closed-loop systems that use rule-based systems to remediate or act on those metrics. Network automation is also increasingly augmented by advanced data analytics, including machine learning (ML), deep learning (DL) and other AI techniques to better understand and predict network usage and security and to optimize its configuration and even topology. When combined with intent-based modeling, such advanced data analytics could enable fully automated closed-loop systems in which the network can automatically measure, assess and react to changing conditions with speed, precision, and insight that isn't possible with manual management methods.

While getting to the eventual goal of a fully autonomous network that is able to self-optimize and even self-heal, the reality today is that most organizations are just starting to apply model-based automation to their networks. Even so, as we indicated earlier in the brief, organizations that choose to go down the automation path can and usually do reap significant business benefits.

From DevOps to NetDevOps

While we've discussed the evolution of network automation techniques, there's also a process and cultural change that's a key part of our journey in network automation. Very often, what happens in the network parallels but lags that in the compute and application domain. For instance, virtualization came first into compute, then storage, and then finally, networking. The benefits of virtualization in the compute world, such as application portability, resource sharing, snapshotting and fast recovery, as well as infrastructure automation, also apply in networking. As networks become more virtual and APIs and programmatic interfaces are exposed, the same journey for application developers will likely show up in networking as well.

DevOps and continuous integration, continuous deployment (CI/CD) practices now dominate most application development lifecycles in the cloud. The ability to continually update and push new code releases, along with a fully automated pipeline that takes care of security checks, application build, staging, testing, and then staged production rollout is the hallmark of most modern software shops. In the networking domain, most enterprises and service providers aren't quite there yet, but many envisage a DevOps equivalent for networking that some are terming NetDevOps. With virtualization and programmatic interfaces into network elements, the vision is that network operators can run an automated pipeline not different from their application counterparts. This pipeline could ostensibly verify the correctness of the network config, check compliance and security, and then be staged, tested (maybe even simulated) before it's pushed into production. And once in production, if issues occur, a simple button push could rollback those changes.

In the networking domain, most enterprises and service providers aren't quite there yet, but many envisage a DevOps equivalent for networking that some are terming NetDevOps.

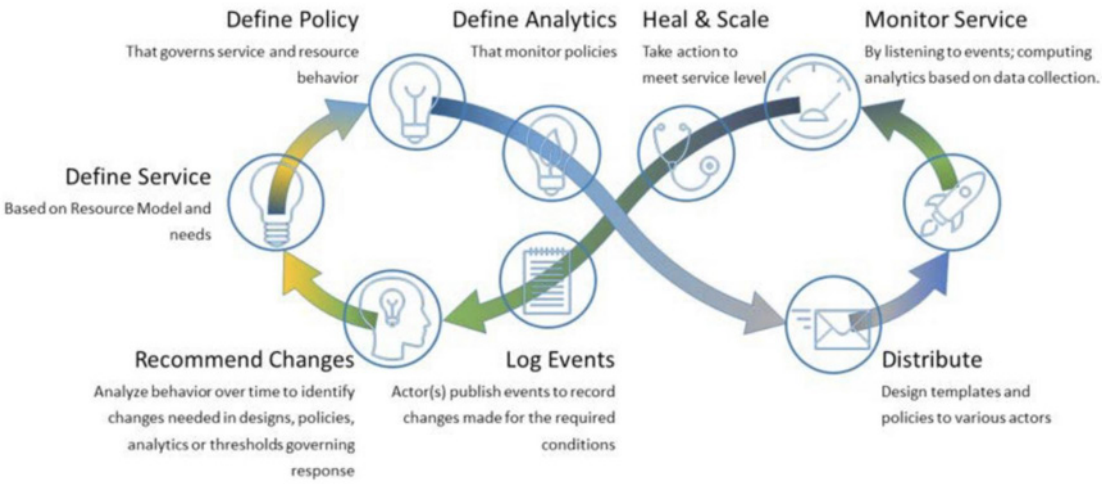
Network Automation Applications and Challenges

We previously noted that network automation was an appealing alternative to more radical network architectural changes because it could be incrementally introduced without disrupting existing systems; another significant benefit is that automation can be applied to any network scenario. While specific tools and programming interfaces will be different, task automation works with:

- Private clouds on data center networks like vCloud, Azure Stack, OpenStack and Kubernetes clusters, configuring virtual network connectivity elements.
- Public cloud network infrastructure from the major IaaS providers, configuring parameters and settings in these networks to allow for connectivity from private locations into virtual private clouds (VPCs) resident in these public clouds.
- Campus LANs and WLANs, setting up network segments, SSIDs, QoS parameters and more.
- Branch office LANs and their WAN connections, creating VLAN segments, triggering the right QoS behavior, laying down security configuration parameters.
- Mobile carrier networks, configuring the radio-area network (RAN), the mobile backhaul transport network, or elements of the core mobile network to set up end-to-end connectivity or QoS.

For example, the two largest U.S. wireless providers, **AT&T** and **Verizon**, along with many foreign carriers, support Linux Foundation's ONAP (Open Network Automation Platform) because it provides a path to full network automation. As one AT&T exec put it, "Together with the community, we are further establishing ONAP as the defacto standard for automation." We should note that AT&T contributed a large portion of the code that formed ONAP (ONAP consists of code contributions from AT&T's ECOMP project, as well as the original Linux Foundation Open-O project). Many may consider ONAP to be more of an orchestration system than a simple automation system, but the end goals are similar though the scope may differ. As

described in the **ONAP Architecture Overview** and illustrated below, its closed-loop automation component can "proactively respond to network and service conditions without human intervention."



ONAP Closed Loop Automation

Source: ONAP Architecture Overview

Other orchestration systems like Open Source MANO (OSM) from ETSI aim to do the same. Within private data centers, **VMware's vCloud NFV**, a version of its popular virtual infrastructure stack optimized for carriers, offers similar closed-loop capabilities. Similarly, OpenStack has a set of network APIs, neutron, that can be automated via OpenStack Heat or third-party orchestration tools to achieve similar results.

Challenges

Network automation looks great in theory; however, implementing it is full of challenges and roadblocks, whether technological, organizational or educational. Indeed, some are generic, applicable to automation efforts in any industry, while others result from the peculiarities of networking. The following hurdles can affect organizations of all sizes and industries:

- **Inadequate skills and expertise** within networking teams, which can be as extreme as not knowing the basics about programming and software development or as fixable as unfamiliarity with a particular language. Even though frameworks and tools like Ansible exist — along with Chef, Puppet, SaltStack and other popular application automation tools — network engineers tend to find them more challenging than their application counterparts in getting started on the automation track.
- **Insufficient time, money, and executive sponsorship of training and career development** are exacerbated by the skills deficit.
- **Siloed organizations** with fragmented management domains and tool choices that make standardization nearly impossible. AvidThink has talked with many enterprises and service providers in which automation may exist in different siloes within the organizations, but they all use different tools and frameworks making it hard to share or integrate across domains.
- **Lack of industry standards** defining common software abstractions and network programming models across vendors and device categories. Network architects don't know how the underlying models, TOSCA, YANG, JSON etc.,

will be provided, and yet they will have to manage across them. Such inconsistency also manifests itself in the different definitions, capabilities, and configuration options vendors have for various device types such as routers. Without a standard, shared model, it's hard to represent hardware as an object that code can manipulate. The **OpenConfig working group** aims to solve some of these issues.

- **Inconsistent or nonexistent programming models and interfaces** across vendors, with CLIs and APIs that don't interoperate. For example, a vendor's CLI might not have commands for manipulating a particular feature or parameter. Similarly, there is often feature inconsistency between a vendor's CLI and its APIs meaning some actions can be done in one, but not the other. There is also a notable difference in programmability across different network domains such that the degree of programmability can vary widely across an organization's fleet of devices. For example, data center switches and routers are rapidly becoming more programmable in response to the work of OCP, ONIE, and vendors like Cumulus to create open hardware and APIs. In contrast, edge routers and WLAN equipment are often limited by proprietary interfaces and management software.
- **Inconsistent telemetry** across vendors, which includes both the type and granularity of recorded parameters. Again, there are some efforts here, and historically SNMP, NetFlow, sFlow, jFlow and other vendor-supported mechanisms have provided a good amount of data, but the inconsistency across vendors creates difficulties in automation. Efforts linked to P4 — a network programming language many view as the successor to OpenFlow — include P4-INT, which focuses on in-network telemetry and could provide a more modern and open standard around which vendors could agree.
- **Inadequate vendor-agnostic automation tools** that make creating fully automated workflows difficult. There isn't a software ecosystem comparable to that for DevOps pipelines that include integrable tools covering code development through testing and deployment.

Fear of Machines Gone Wild

Recent high-profile outages at some major cloud operators and online services have also increased the trepidation IT and network managers feel about putting their critical network systems on autopilot. Runaway automation gone bad is a worst-case scenario since it can quickly spread a critical mistake that once would have affected a single equipment rack or data center across an entire enterprise network.

The likelihood of such catastrophic automation errors has been exaggerated by headline-grabbing incidents such as the Google Cloud outage in June 2019)that knocked several high-traffic services such as YouTube, G Suite and Snapchat offline for several hours. As **Google detailed in an incident postmortem (emphasis added)**,

> "In essence, the root cause of Sunday's disruption was a configuration change that was intended for a small number of servers in a single region. **The configuration was incorrectly applied to a larger number of servers across several neighboring regions, and it caused those regions to stop using more than half of their available network capacity.** The network traffic to/from those regions then tried to fit into the remaining network capacity, but it did not. The network became congested, and our networking systems correctly triaged the traffic overload and dropped larger, less latency-sensitive traffic in order to preserve smaller latency-sensitive traffic flows, much as urgent packages may be couriered by bicycle through even the worst traffic jam. **Google's engineering teams detected the issue within seconds, but diagnosis and correction took far longer than our target of a few minutes.**"

Such incidents give executives and other decision makers a false impression of the fragility of network automation. While rogue automation code can amplify the damage of some mistakes, the code itself is seldom the cause of outages.

Far more common are mistakes by human network administrators using CLIs to misconfigure or incorrectly update equipment; they even admit as much. In a 2016 survey of network professionals, **45% say that human error causes all or most network outages**, with a full 97% acknowledging human error as a cause of at least some network downtime. Furthermore,

such human-caused problems are common, with 30% saying that network changes cause an outage or performance issues several times a month or more. Sadly, network operators aren't even particularly good at monitoring their networks, with 60% saying that some, if not all network outages are first reported by users, while only 6% have monitoring systems that can predict 90% or more of their networking problems.

Another survey also found that **human error is the top factor negatively affecting network reliability**, outpacing both security and network complexity. Indeed, Google's postmortem is ambiguous on the root cause of its mega-outage, but it is highly likely that the incorrect application of configuration code was made by a human operator pushing out the update. In the long run, handing network controls over to automation improves reliability, whereas trusting humans inevitably leads to more mistakes.

The ideal of an all-encompassing automation tool that works across every network environment and piece of installed equipment **isn't realistic for most organizations**.

Recommendations

Done well, network automation can significantly improve network performance, reliability, availability, and security, while reducing administrative overhead and the time required to fix problems, add services, and incorporate new applications. However, automation can only realize its potential through careful planning of design and deployment to achieve a system that works across the entire mix of enterprise networks and that is understood and embraced by IT staff. The following are some recommendations and best practices that can help turn these goals into reality.

- **Build a system that normalizes automation across network domains with a model-based approach** that has one or more, if well integrated, automation tools that cover all network environments: the data center, campus WLAN, WAN, remote sites and edge/IoT networks. Model driven means a system that treats key network elements as software abstractions that can be programmatically configured and integrated via APIs. Such a model-based, procedural system sets the foundation for the ultimate goal of intent-based network automation in which operators specify network behavior and outcomes and let an intelligent system translate these business and application goals into specific network topologies, parameters and configurations for individual equipment.
- **Identify the limitations and constraints for each platform when mapped to a model-based approach.** For example, the ideal of an all-encompassing automation tool that works across every network environment and piece of installed equipment isn't realistic for most organizations. Instead, they must piece together automation capabilities available within their existing network management consoles with selective add-ons for different domains. However, such a heterogeneous scenario requires IT to identify various integration points between network management domains along with links to other systems like CMDBs and software-build processes. Once these touchpoints are understood, they often require additional work to integrate automated actions between domains and, for example, keep CMDB catalogs updated with automation-triggered changes.
- **Align automation with cross-domain visibility (monitoring) and performance management automation platforms.** Two of the critical integration points between automation tools and existing network systems involve monitoring and performance measurement. Intent-based automation is a closed loop system in which the automation engine dynamically responds to changing network conditions, whether workload fluctuations, equipment failures, or even cyberattacks. As such, intent-based automation requires real-time feedback from equipment telemetry and other monitoring platforms to populate models, which often use machine or deep learning, to trigger changes in network configuration based on changing conditions. Thus, a complete network automation regime requires tight integration between automation and monitoring tools.

- **Engrain automation into operational processes; it can't be a one-off, set-and-forget task.** There's a cultural and organization element to network automation that is just as important as the underlying technology since without both staff buy-in and procedural embodiment, automation can turn from a force multiplier into an added source of overhead and frustration. Organizations can't afford to only focus on Day 0 system design activities, since Day 1 deployment and implementation and Days 2 and beyond operations are equally essential and require staff acceptance and entrenchment into their daily routine.
- **Skills training for team members.** While listed last, as the previous point makes clear, successful automation projects require active employee support, engagement, and understanding. Although the goal is a system that eventually, mostly, runs itself, like any robotic tool, it requires humans to design and take over in situations the algorithms can't handle. Given the complexity and conceptual abstractness of sophisticated automation software, such staff involvement requires structured training and the time required to learn, test, and become acclimated to an automated environment.

As the infrastructure required to run and scale modern applications gets more complicated and voluminous, it renders manual systems management processes obsolete, crushed under the weight of thousands of devices and millions of monitoring events. As in countless industries before, process simplification, systematization, and automation are the only way to operate enterprise- and cloud-scale infrastructure and networks. Next-generation automation software that treats network devices and parameters as programmable objects and that responds to high-level, model- or intent-based commands is the key to operating massive, distributed, heterogeneous networks efficiently, reliably and securely.

AvidThink, LLC
1900 Camden Ave
San Jose, California 95124 USA
avidthink.com

Rev A