The Netivus Manifesto: Making Collaborative Network Management Easier for the Rest of Us

Joseph Severini
Carnegie Mellon University, USA
jseverin@andrew.cmu.edu

Radhika Niranjan Mysore
VMware Research, USA
rniranjan@vmware.com

Vyas Sekar
Carnegie Mellon University, USA
vsekar@andrew.cmu.edu

Sujata Banerjee
VMware Research, USA
sujatab@vmware.com

Michael K. Reiter
Duke University, USA
michael.reiter@duke.edu

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

Abstract

We study operational issues faced by Small and Medium Enterprise (SME) network owners and find that SME network management practices have stagnated over the past decade, despite many recent advances in network management. Many of these advances target hyperscalers and ISPs and cannot be directly applied to SME networks that are operated with vastly different constraints. In our work, we outline these constraints and explain how they impact challenges around debugging, namely: representing, reproducing, and remediating network problems. This article takes a fresh look at these challenges in the light of SME practices around collaborative debugging and presents a roadmap aimed to help resolve SME operational issues quickly.

CCS Concepts

• **Networks** → **Network manageability**.

Keywords

Collaborative Debugging, Manageability

1 Introduction

Networking research has made progress on multiple fronts with technologies such as software-defined networking, network function virtualization, new transport algorithms, new directions in network verification and testing, and even hardware capabilities. Unfortunately, most of these advances are driven by, and are largely restricted to, the "giants" (e.g., hyperscaler datacenters, large ISP networks).

An open question is whether these advances in network management matter to *the rest of us*; i.e., small and medium enterprise (SME) networks. These networks are little understood, much less served, by the research community. Anecdotal evidence and surveys [18] suggest that these networks do not have the capital/operating budgets for adopting new technologies and implementing "DevOps" workflows associated with these developments. Instead of being "software-driven", where network operators can develop custom management toolchains, these networks are *vendor-driven* and reliant on vendor-specific tools. Managing a network composed of devices from multiple vendors is hard, because common tooling (beyond a few like `ping`, `traceroute`) is usually unavailable.

Given these vendor-driven constraints, a natural recourse available to SME operators when they encounter unexpected issues is the wisdom of others who have faced similar challenges. Indeed, there is a rich tradition of operator events [10], mailing lists [8], and more recent StackExchange-style online forums [3], by which SME operators engage in *collaborative debugging* of network management issues. While these forums bring together a community of operators to collectively share their experiences, pain points, and wisdom, these platforms are neither smooth nor perfect for trouble-shooting and often leave problems unsolved (Section 2).

This paper presents the **Netivus Manifesto**[1]—*Given that operators are already attempting collaborative debugging, can we make it easier and more effective?* We argue that this is a more pragmatic approach to improve the status quo for SME networks, in contrast to new paradigms or software-defined agendas. Our manifesto was informed by analyzing 100 operation-related posts to the Network Engineering Stack Exchange (NESE) forum. Only 36 of the 100 were answered within 3 days, and 35 questions were never resolved. Operators struggled to describe their problem, needing a median of 1.1 days to create a post that contained the information collaborators required. Collaborators needed a median of 1 week to understand the problem before suggesting fixes. The poor resolution and delays were not due to a lack of collaboration; 80% of unresolved questions received ≥ 2 comments, and 75% received at least one answer.

Our findings point to three key challenges that operators face when **representing**, **reproducing**, and **remediating** network issues (we call these the "**3R**" steps) using collaborative debugging. First, operators must be able to describe the issue sufficiently to collaborators; we find that operators today have no shared language or practices for specifying network problems and most (81%) question posters rely on their collaborators to help them do so. We call this the **representation** challenge with collaborative debugging. Second, collaborators must construct mental models of the problem scenario with just the problem text and configuration files provided; our own attempts (Section 4.3) show this is hard to do, and the data shows collaborators can take days to form a sufficient model. This problem of creating a shared network model between problem posters and collaborators is the **reproduction** challenge. Finally,

[1] A tongue-in-cheek reference to a Festivus for the rest of us [28].

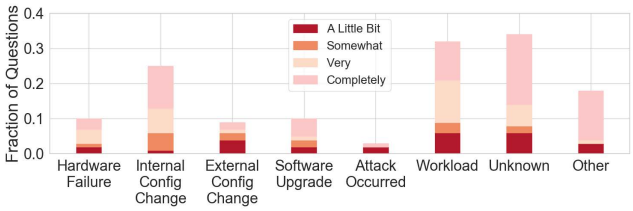

Figure 1: Distribution of triggers for operational issues

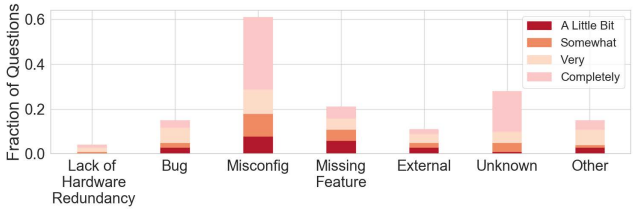

Figure 2: Distribution of diagnosis of operational issues

since collaborators must rely on their intuition and experience to identify root-causes and suggest fixes, they often ask the problem posters to test potential fixes on the live operational network. Doing so is daunting even for the most well-managed and instrumented networks [20]. We call this heavy reliance on collaborators' intuition for root causing and identifying fixes as the **remediation** challenge.

Our contribution is in formulating these challenges and outlining a vision for a new collaborative debugging roadmap for SME operators. Although we do not have answers on how to address these challenges, we identify promising starting points. We hope the community can collectively come up with solutions to open questions we raise and address the needs of this under-served community.

2 Background and Motivation

To better understand the constraints and challenges faced by SME network operators, we analyzed operational issues posted on Network Engineering Stack Exchange [3] forum (NESE), for the past 7 years. Users use this forum to post questions and get help with their network-related queries, and experts weigh in by posting comments (for clarification and criticism [2]) or answers (for answering the question [4]). Questions are edited by the user who posted them (to include additional information, clarify, etc.) or by others (e.g., to correct minor mistakes [13]). The original poster can mark the answer that best serves as a solution.

Methodology. We sampled 1200 posts from this forum and (manually) identified a subset of 100 posts relevant to debugging live SME networks. We then used a combination of approaches to analyze these posts. First, we manually scanned the posts to gather anecdotal evidence of the types of problems SMEs face. Second, we analyzed the posts quantitatively to get a sense of the degree and success of collaboration, e.g., using measures like number of comments and answers and time to resolution. Third, we used a crowd-sourced approach using an Institutional Review Board (IRB) approved survey to help classify the posts across various axes.

The survey participants included graduate students, researchers, and experts from NANOG [7]. Each survey participant was presented a set of posts, for which they classified the types of problems reported, their triggers, and their root causes into predefined categories (raw results available here [11]). These questions were inspired by prior work on analyzing network failures [20, 29, 37]. The x-axis of Figures 1-2 shows the categories used for triggers and root causes. The participant was also asked to classify the post as a connectivity problem and/or a performance problem, and/or something else ("other" category). Respondents used a Likert scale [25] to specify the extent to which the problem fits into each category, from "not-at-all" to "completely".

2.1 Complexity of SME operations

While this analysis is preliminary, it sheds light on a number of unique challenges SME network operators face.

- SME networks are largely unstructured and are grown organically over years. Many have a high degree of heterogeneity with devices from multiple vendors. Therefore, most SMEs cannot limit operations to uniformly using a single management toolchain from a vendor for their entire network, unlike hyperscalers [22, 35, 38]. From our reading of the posts, SME operators might not even have a topology map of the network unlike other well-managed settings [30, 34], nor expertise with vendor solutions.
- Most SME network operations are not automated. Our findings are consistent with the 2019 DevOps survey [18] that found that only 10-20% of participants used automated solutions for troubleshooting and remediation. Without shared and widely-understood tooling and debugging solutions, SME operators lack a common language to discuss issues, making it hard to adopt state-of-art techniques to analyze network issue reports like NetSieve [33].
- Figure 1 shows that SME operational issues can be triggered by a diverse variety of events. Issues can seem to crop up randomly (e.g., unknown triggers), while others are caused by workload or configuration changes (e.g., due to introduction of new network hardware or application upgrades that break the network). Many triggers are identified only after significant back and forth with collaborators. The fact that many issues have unknown triggers points to a lack of tools that can provide comprehensive operational visibility and monitoring in SME environments.
- Figure 2 shows that roughly half of the operational issues are attributed to misconfigurations; others are mostly missing features or bugs. Diagnosing misconfigurations has been tackled by a large body of literature, including on verification [17, 23, 24]. They rely on complete knowledge of configurations from a network, which is often unavailable in SME network environments.

Our analysis also revealed that recent advances in network automation, monitoring, and verification are not widely used by SME networks. As such, they are restricted to using canonical tools like `ping`, `tcpdump`, `traceroute`, `snmp`, among others.

Anecdotally, we also found that often the only alternative to collaborative debugging for SMEs is expensive service engagements with the relevant device vendors (e.g., [9]).

2.2 Challenges of collaborative debugging

Our quantitative analysis of posts in NESE showed problems with current collaborative debugging practices available to SME operators today. By reading 1000 other posts, we also verified that these

Figure 3: Resolution time across posts (35 unresolved)

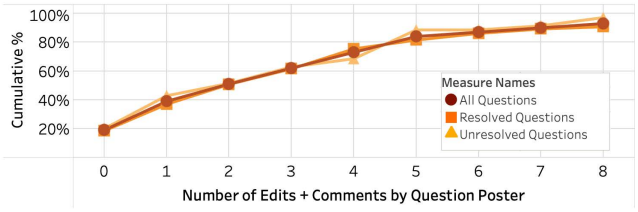

Figure 4: Distribution of question updates across posts

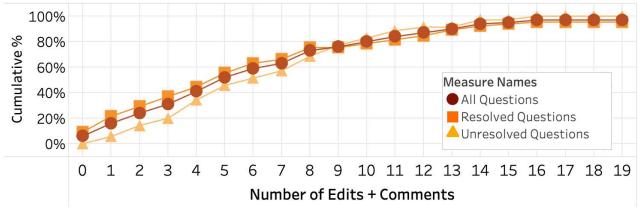

Figure 5: Distribution of number of updates across posts

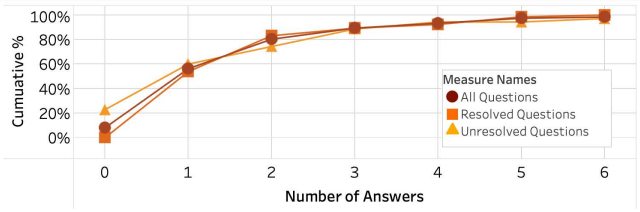

Figure 6: Distribution of number of answers across posts

challenges exist for posts that go beyond debugging, e.g., that seek help on network deployment and configuration. We do not present those results because we did not manually categorize these posts as we did the 100 posts around debugging live networks. We present our observations below.

1. Problems take a long time to resolve or remain unsolved. Only 36% of posts were resolved within 72 hours and 46% of posts within a week. Also, 35% were never resolved (Figure 3). While we do not have conclusive data whether the operators were able to maintain critical network functionality during the debugging period, this suggests a non-trivial amount of downtime for SMEs.

2. Initial problem descriptions are often incomplete. 81% of questions were edited at least once (Figure 4), typically in response to clarifications requested by collaborators.[2] Posts that were resolved and posts that were not resolved showed similar trends. Clarifying the question took a median time of 16 hours, again suggesting significant time during which some service may be down.

A question titled "Access port config" [1] illustrates this issue. The poster operates a network (see Figure 8) with a star topology, consisting of several layer-3 switches, VoIP phones, and a VoIP server on VLAN 10. When the operator connected a new switch to the central switch in the star-topology, the VoIP server became unreachable (from some unspecified point in the network), but the phones continued to work. The poster initially posted this problem without topology or configuration information. Once the collaborator pointed out the missing information, the poster was able to supply them within a couple of hours, after which the issue was resolved. The root-cause, an IP address conflict on VLAN 10, was apparent from the configuration files.

3. There is significant discussion, yet issues are unresolved. One possible hypothesis for Observation 1 is that there is little engagement for many of these unresolved questions. However, we find this is not the case. Collaborators engaged with the operator in more than 80% of posts (Figure 5) (both comments and significant edits) and we see no difference between the activity for resolved vs. unresolved posts. This leads us to believe that lack of collaborator engagement is not a cause for posts going unresolved.

4. Proposed fixes might not work. We found that in nearly all resolved questions, the poster had to test it on their live network. 44% of questions received more than one answer (Figure 6), suggesting significant operator effort—not all answers posted by collaborators work and the burden of identifying the right ones is left to the poster. This trend is similar across resolved and unresolved posts.

3 Research Challenges in Netivus

In the last section, we outlined constraints of operating SME networks and challenges of collaboratively debugging them. In this section, we revisit the challenges in the framework of the 3R steps, by viewing them against the requirements of each step.

3.1 Representation

To get help, operators must first describe (*represent*) the issue to collaborators. Observation 2 in Section 2.2 shows that doing so can be hard. Ideally, operators must describe their problem **sufficiently** but **concisely** and in a **privacy preserving** manner. To be sufficient, the problem representation must supply all aspects required to model the problem, both the network behavior and the observed error. But to be concise, only the right subset of the network must be described; naively providing the entire network configuration may distract collaborators and complicate the debugging process. To be privacy-preserving, operators must not reveal sensitive data (e.g., IP addresses, personally identifiable information (PII), firewall rules) about their network when representing the problem. An example of how operators can inadvertently reveal sensitive information is given in Figure 9 in Section 4.3.

Challenges. To create a problem representation that meets all three criteria, operators face several challenges. First, they must be able to describe their topology, even when they might not readily have a topology map of the network they operate; this usually happens when the operator inherits an already running network. Second, as

[2]This is not counting edits to question titles or tags.

they construct this topology description, they must identify any missing pieces that are needed to create a sufficient description. Third, instead of just sharing a sufficient description, they must decide which parts of the topology description are relevant, to avoid overwhelming collaborators. This is incredibly hard to do, and we find some operators trying to diagnose the problem to choose such a subset *before* they post their question to get debugging help! Lastly, it is incredibly hard for operators to measure or describe their workload characteristics. Measuring network traffic might alter live networks' performance characteristics (reducing availability of their applications). Describing it might require compromising sensitive private configurations associated with applications they use. Given these challenges, unsurprisingly, many posts we analyzed had little to no description of workloads.

3.2 Reproduction

To debug the problem, collaborators must construct an **accurate** mental model (*reproduction*) of the problem and network behavior. Observation 3 in Section 2.2 reveals that constructing such a model is hard. Collaborators need details of connectivity and QoS across relevant network layers. It also helps to know of events that might have triggered the issue. Many such details might not be readily available in the problem description (Observation 2, Section 2.2).

Challenges. Even when collaborators are supplied with a good problem description, bugs (which cause 15% of posts we analyze), missing features (which cause 21% of posts we analyze) and default settings in vendor devices (counted as part of misconfiguration issues that comprise 50% of posts) might change network behavior in unexpected ways. To identify these hidden elements, collaborators might ask operators to run simpler, well known tests (e.g., `ping`, `traceroute`, specific workloads) on their live network and report outcomes (e.g., interface counters). While this process often helps to clarify network behavior, it also consumes time and effort both on the part of the operator, who must run these tests without disrupting their network, and the collaborator who must work with the operator until they can construct an accurate mental model.

3.3 Remediation

Ideally, at the end of the debugging process, operators and collaborators know the root cause of the problem and identify modifications to the network that will resolve it (*remediate*). The root cause must be **accurate**, and modifications easy to apply **quickly**. Ideally, an inexperienced operator seeking help is not expected to try out multiple *different* modification suggestions on a live network (Observation 4); applying such changes and rolling back those that do not work is hard even in well-managed networks [20].

Challenges. In our experience, while identifying the root cause of a network problem is incredibly hard, coming up with remedying modifications can be an art. To ensure that a modification will work, collaborators might need accurate knowledge of causal interactions between devices and protocols in the network, even some that are not part of the original problem description! Under-represented aspects of the network can especially hurt; if the operator is unlucky, device defaults, bugs, or missing features can get in the way of remediation *and* create new problems. Collaborators might also need to understand the original intent of the network operator to suggest feasible fixes. For instance, in one of the posts, a collaborator suggested using BGP to enable routing, only to learn that pure L2 solutions were desired by the operator [6].

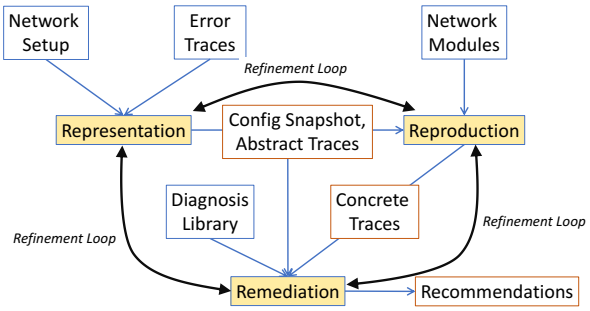

Figure 7: Netivus workflow. 3R steps highlighted.

3.4 Summary

When we view collaborative debugging in the framework of the 3R steps, we observe that missing information at every step prevents operators and collaborators from having a complete, shared picture of network behavior. For example, operators might not have full knowledge of network topology, let alone shared language and practices to describe a problem completely, and collaborators might be in the dark regarding the operators intent and network specifics. It is this fundamental challenge that we want to remedy with Netivus.

We envision that it is possible to build tooling that helps identify what might be missing at every step rather than solely depending on collaboration to do so. As the operator and collaborator fill in the missing information flagged by Netivus, they can refine their shared mental model of the network and get a collective understanding of the problem. We also hope such tooling can help standardize the collaborative process. It must help operators represent network problems with details that collaborators commonly look for, and allow collaborators to focus on parts of the problem that tooling cannot help supply. This intent shapes our preliminary thoughts on improving the collaborative debugging experience.

4 Preliminary Approach

Our mental model of collaborative debugging viewed in the framework of the 3R steps is shown in Figure 7. We use this figure to also capture the Netivus workflow: a workflow that augments each step with software modules (tooling) to aid the collaborative debugging process. The figure shows the inputs and outputs of the software modules. We expect the network information (e.g., network setup, configuration snapshot) used in prior steps of the workflow will need to be refined iteratively by using information from subsequent steps (e.g., concrete traces, diagnosis recommendations), and that such refinement will further clarify the shared mental model between operators and collaborators. For example, if the concrete traces from the reproduction step shows that relevant network devices are missing from the problem description, operators might be able to supply missing configurations of these devices as input to the representation module, and that in turn will result in a better reproduction. We capture this process of refinement using bidirectional arrows between the three steps (labeled 'refinement loops').

In this section we describe existing work that we can use as starting points to implement the Netivus vision. We then posit how

refinement will work, and end with a sketch of how the Netivus workflow might help debug a problem described on NESE.

4.1 Starting points

Representation. To help operators create complete and useful descriptions of their problems, we envision a *representation* module that enables operators to describe their network using topology diagrams, device configurations, any available workload traces (e.g., `pcap`), and error traces (e.g., interface counters). From the operators' description, the module must construct a network model consisting of relevant parts of the configuration files that describe its connectivity, QoS, and error characteristics (configuration snapshot); in addition, it must produce an abstract trace that describes the error in this network model. If it cannot do so, it must identify missing information that it needs to construct such a model (or trace). The representation module must interface with the reproduction and remediation modules to aid refinement, and it must be able to flag missing information identified in any step to the operator.

There are tools like Batfish [17], that can parse configurations to construct a layered view (L1-L4) of how packets flow across a topology, that we can leverage for building the representation module. Tools like Netconan [5] are already able to anonymize IP addresses in configurations, a feature that can be useful to build privacy-preserving problem representations. For developing workload representations, we can look to synthetic traffic generators [21, 36] that have long been used for network testing. We think that an interesting question around workload generation is whether Generative Adversarial Network (GAN)-based traffic generation can be leveraged as a privacy-preserving alternative to sharing raw traces [26]. We hypothesize that the GAN could potentially be trained using recent packet logs (e.g., from router SPAN ports) triggered during the event.

Reproduction. This module takes the output of the representation module along with any vendor-supplied modules, such as device images, and creates an emulation of the problem to the extent possible. If the error is reproducible, then the module must make it possible to extract the concrete error trace and supply it to the remediation module. If the error is not reproducible, the module must identify information it might be missing to recreate the error trace, and feed this information back to the representation module (refinement).

There are a wide range of simulation and emulation tools such as CrystalNet [27], ns3 [32], and GNS-3 [19] that can be leveraged as the starting point for the reproduction module. Some, like GNS-3, also allow users to input images to improve emulation fidelity. For devices without images or configurations, we imagine we might be able to synthesize their behavior models using tools similar to Alembic [31], and use these models for reproduction.

Remediation. The remediation module will take the output of the reproduction module and a diagnosis library that might, for instance, provide a canned set of candidate hypotheses to help narrow down the potential root causes and solutions. It must then not only help localize the network issue, but also recommend modifications to the network to ideally resolve the issue. To make such recommendations, the module might need information about parts of the network that are not impacted by the issue. Therefore, the remediation module must interface with the representation and reproduction modules to clarify aspects of the network before suggesting modifications.

Network verification techniques [17, 23, 24] already help localize certain issues from configuration and dataplane snapshots; if the representation module can supply similar snapshots, these tools can be leveraged in the remediation module for fault localization. We also imagine that mining vendor manuals/best practices [12] and NESE posts using techniques from NetSieve [33] might provide initial lists of hypotheses. If the operator is able to supply the set of configuration changes they made to the network before running into the problem, techniques like differential provenance [15] can be used to identify relevant configuration changes that might be triggers. Once problematic configuration snippets are identified, we speculate we can use techniques like NetComplete [16] to generate alternative configuration snippets that might help resolve the problem.

4.2 Refinement Loops

We have so far described the Netivus modules. We now describe how operators, collaborators, and the modules themselves might interact to refine network representation, issue reproduction, and remediation recommendations. We believe that these refinement loops are key to improving collaborative debugging resolution rates and outcomes.

We envision that an operator has access to a private instance of the representation module that they iterate with to supply missing information, until the module can construct a complete representation of the network and the corresponding abstract error trace. The module must then remove any private information and supply the operator with a representation that they can share with collaborators, perhaps by using an online version of the representation module. To be effective, the representation module must reduce the amount of collaboration needed to arrive at a complete, privacy-preserving problem description.

The online version of the representation module will interface with an online reproduction module to recreate the issue. If the issue is reproducible, then these modules must interface and iterate until a minimal subset of the network that is sufficient to reproduce the issue is identified, by gradually whittling away parts that do not impact the issue. We believe that creating such a minimal representation for collaborators will help reduce problem complexity and might improve their chances of arriving at a remediation quickly.

If the reproduction module is unable to reproduce the issue, it must suggest other workloads and tests the operator can try in order to localize the mismatch between the real network and the module's network model. The operator can then supply the outcomes of these tests to the representation module to further refine the network model that the reproduction module uses. By suggesting workloads and tests that are routinely requested, the reproduction module can reduce the number of unknowns that collaborators will encounter.

If the reproduction is successful, the remediation module must generate a ranked list of candidate hypotheses that are most likely to cause the issue. The reproduction and the list of hypotheses are starting points for collaborators' investigations. Once collaborators identify changes that might remedy the problem, they can test them using the reproduction module. This ensures that disruptive modifications can be weeded out before presenting them to the operator.

On the other hand, if the collaborators need more information to debug or remedy the issue, they can fall back to asking the operator to supply those inputs to the representation module and restart the Netivus workflow.

Abstract Trace
(1) Packet to VoIP server sent from Cisco 3750 switch
(2) No response

Table 1: Abstract events that provoke error symptoms.

Concrete Trace
(1) Packet to VoIP server sent from Cisco 3750 switch
(2) Packet arrives at layer-3 switch connecting VLANs 5 and 10 (the new Cisco 3850x switch)
(3) New Cisco 3850x switch floods packet out all interfaces except the one that the packet arrived one
(4) All flooded packets dropped

Table 2: Concrete events that lead to error symptoms.

Potential Root-Cause	Fix
IP address conflict	Change one of the IP addresses
Server is down	Restart server
ACL drops traffic	Remove rule that drops traffic
Different VLAN	Switch VLANs
Cable malfunction	Replace cable

Table 3: Root-cause hypotheses and potential fixes

4.3 Collaborative debugging with Netivus

We now present a mental sketch of what it might look like to apply collaborative debugging with the Netivus workflow to a problem described in a post titled "Access port config" [1] described in Section 2.2. Recall that the network had a VoIP server, which became unreachable when the network operator connected a new switch. We now describe the 3R steps for debugging this problem using Netivus.

Representation step. Using the configuration files inputted by the operator, the representation module creates a configuration snapshot (Figure 9) that describes the parts of the configuration file that are needed by the network emulator in the reproduction step. This snippet is constructed by removing personally identifiable information (PII) and other information not needed for emulation, such as logging and management settings, from the configuration files supplied by the operator. The representation module also generates the abstract trace (Table 1) concisely describing the error, and some information of the action that led to the error, from the original error description.

Reproduction step. The reproduction module uses the configuration snapshot and abstract trace to generate a minimum, sufficient emulation of the problematic network. To do so, it uses available device images and configures them based on the configuration snapshot. It then applies the actions in the abstract trace and checks if the described problem occurs in the emulator's event log. If it does, then the snippet in the log associated with the error is captured as the concrete trace, in addition to the information about the actions that led to the error. Table 2 shows the concrete traces we constructed. Since we did not have an actual emulation, we instead used a network model we constructed using the configuration snapshot shown in Figure 8 and the abstract trace to come up with the concrete trace.

Recall that the representation and the reproduction modules iterate to refine the problem representation and make it more concise. By removing network elements that do not impact the error condition, this refinement loop eliminates the dotted lines shown in Figure 8.

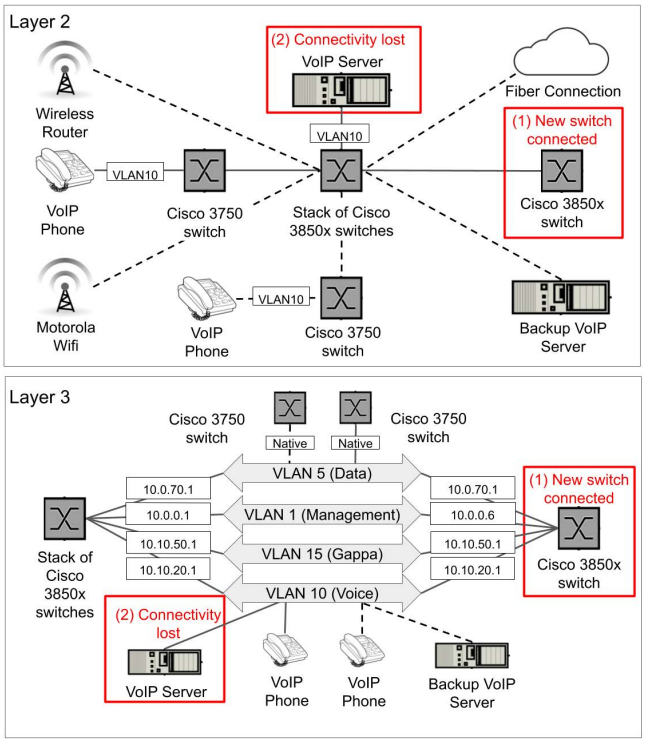

Figure 8: Network representation: Dotted links were removed for the minimal, sufficient representation.

Remediation step. The remediation module uses a list of hypotheses from sources like Cisco network troubleshooting guide [12] as a starting point for the diagnosis library. We can further refine some of their high level suggestions (e.g., configuration problem, physical layer issues) with more concrete ideas (e.g., server is down, ACL drops traffic, different VLAN, cable malfunction), drawing from the root-causes we find from the posts we analyzed. We expect that the list of hypotheses will be similarly refined continuously in the Netivus platform. The remediation module can also use verification tools (e.g., [17]) to analyze configurations and generate intermediate network models like Figure 8.

Both the hypotheses from the Cisco troubleshooting guide and running reachability tests on the network model point to an IP address conflict as the most likely root-cause. The refinement loop with the reproduction module, and the verification tools, can help pinpoint the exact IP address conflict. The remediation module suggests this as the most likely issue among the hypotheses identified in Table 3. It also suggests potential fixes, again mined from the posts we analyzed. Finally collaborators and operators can use the reproduction module to verify this is the simplest resolution.

5 Open Research Questions

Our journey to understand the problems faced by SME operators and identify what might help has brought to light several open questions. We conclude the Netivus manifesto by outlining them.

Usability. All of the starting points described in Section 4 require considerable network, software, and hardware expertise before one can use them. For network operators with limited resources, this

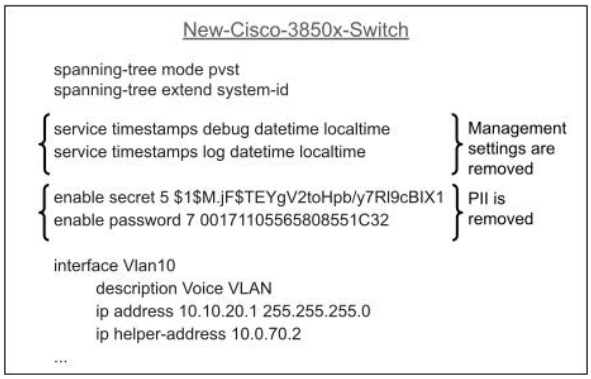

Figure 9: Snippet of the configuration snapshot.

presents a barrier to adopting some of these tools. There is a need to rethink the software interfaces that many of these tools provide to be usable by SME operators. We anticipate there is also much to be learned from the experiences of the computer supported cooperative work (CSCW) community to inspire broader adoption.

Interfaces. Stitching together various available stand-alone tools to realize the Netivus manifesto is not easy. Each of the tools described will require modifications before they can interface with other tools. For instance, there is no way today to plug Batfish network representations or behavioral models generated by Alembic into GNS-3. While none of these tools were developed with such interactions in mind, we believe that going forward, it will be critical to have a principled approach to defining interfaces for each of these tools to increase adoption and impact.

Missing Pieces. Even if the usability and interfaces of these tools are improved, we will be left with the missing pieces and open questions described in Section 3. For instance, how can one ensure a representation sufficiently captures the details required for debugging the problem? How can operators capture workload characteristics faithfully across a network? We find that many of these questions challenge the assumptions made by current tools around homogeneity of deployments and software defined nature of networks.

A broader manifesto. We focused on troubleshooting in SME networks since this was a major source of pain points for users on NESE. Our analysis of the NESE forum, however, also reveals numerous struggles with management operations, superfluous devices or protocols, device/network setup, and upgrades. We believe there is a rich opportunity in treating these topics as first-order research problems; e.g., network debloating to cull out unnecessary components, ML-driven configuration optimizations (e.g., [14]), or better versioning or snapshotting tools for SMEs.

Acknowledgments

This work was funded in part by the NSF/VMware Partnership on Software Defined Infrastructure as a Foundation for Clean-Slate Computing Security (SDI-CSCS) program under Award No. CNS-1700521.

References

[1] Access port config. https://networkengineering.stackexchange.com/questions/45650/access-port-config.

[2] Comment everywhere. https://networkengineering.stackexchange.com/help/privileges/comment.

[3] Explore our questions. https://networkengineering.stackexchange.com/.

[4] How do I write a good answer? https://networkengineering.stackexchange.com/help/how-to-answer.

[5] Intentionet network configuration anonymizer. https://github.com/intentionet/netconan.

[6] Multihomed internet edge without BGP. overlapping routes, a viable design? https://networkengineering.stackexchange.com/questions/19763/multihomed-internet-edge-without-bgp-overlapping-routes-a-viable-design.

[7] North American network operators group. http://nanog.org.

[8] North American network operators group mailing list. https://mailman.nanog.org/mailman/listinfo/nanog.

[9] Qos woes - managed ip vpn. https://networkengineering.stackexchange.com/questions/2427/qos-woes-managed-ip-vpn.

[10] Réseaux IP Européens (RIPE) Meetings. https://www.ripe.net/participate/meetings/ripe-meetings.

[11] The-netivus-manifesto-data. https://github.com/fretbuzz/The-Netivus-Manifesto-Data.

[12] Troubleshooting tcp/ip. https://www.cisco.com/en/US/docs/internetworking/troubleshooting/guide/tr1907.html.

[13] Why can people edit my posts? How does editing work? https://networkengineering.stackexchange.com/help/editing.

[14] O. Alipourfard, H. H. Liu, J. Chen, S. Venkataraman, M. Yu, and M. Zhang. Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 469–482, Boston, MA, Mar. 2017. USENIX Association.

[15] A. Chen, Y. Wu, A. Haeberlen, W. Zhou, and B. T. Loo. The good, the bad, and the differences: Better network diagnostics with differential provenance. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, pages 115–128, New York, NY, USA, 2016. Association for Computing Machinery.

[16] A. El-Hassany, P. Tsankov, L. Vanbever, and M. Vechev. Netcomplete: Practical network-wide configuration synthesis with autocompletion. In *Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation*, NSDI '18, pages 579–594, USA, 2018. USENIX Association.

[17] A. Fogel, S. Fung, L. Pedrosa, M. Walraed-Sullivan, R. Govindan, R. Mahajan, and T. Millstein. A general approach to network configuration analysis. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, NSDI '15, pages 469–483, USA, 2015. USENIX Association.

[18] D. Garros. The state of network operation through automation / NetDevOps survey 2019. http://blog.networktocode.com/post/state-network-operations-netdevops-survey-2019/, 28 Apr. 2020.

[19] GNS3 Technologies Inc. GNS3 | The software that empowers network professionals., 2016.

[20] R. Govindan, I. Minei, M. Kallahalla, B. Koley, and A. Vahdat. Evolve or die: High-availability design principles drawn from googles network infrastructure. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, pages 58–72, New York, NY, USA, 2016. Association for Computing Machinery.

[21] Ixia. Ixia traffic generator. https://www.ixiacom.com/.

[22] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. B4: Experience with a globally-deployed software defined wan. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 3–14, New York, NY, USA, 2013. Association for Computing Machinery.

[23] P. Kazemian, G. Varghese, and N. McKeown. Header space analysis: Static checking for networks. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI '12, page 9, USA, 2012. USENIX Association.

[24] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey. Veriflow: Verifying network-wide invariants in real time. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, nsdi'13, pages 15–28, USA, 2013. USENIX Association.

[25] R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 140:1–55, 1932.

[26] Z. Lin, A. Jain, C. Wang, G. Fanti, and V. Sekar. Using GANs for sharing networked time series data: Challenges, initial promise, and open questions. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, pages 464–483, New York, NY, USA, 2020. Association for Computing Machinery.

[27] H. Liu, Y. Zhu, J. Padhye, J. Cao, S. Tallapragada, N. Lopes, A. Rybalchenko, G. Lu, and L. Yuan. Crystalnet: Faithfully emulating large production networks. In *SOSP '17 Proceedings of the 26th Symposium on Operating Systems Principles*, pages 599–613. ACM, October 2017.

[28] M. Locker. The origins of Festivus, the festival for the rest of us. https://time.com/4617553/festivus-holiday-origins-seinfeld/, 23 Dec. 2016.

[29] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. N. Chuah, and C. Diot. Characterization of failures in an IP backbone. 4:2307–2317, 2004.

[30] J. C. Mogul, D. Goricanec, M. Pool, A. Shaikh, D. Turk, B. Koley, and X. Zhao. Experiences with modeling network topologies at multiple levels of abstraction. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 403–418, Santa Clara, CA, Feb. 2020. USENIX Association.

[31] S.-J. Moon, J. Helt, Y. Yuan, Y. Bieri, S. Banerjee, V. Sekar, W. Wu, M. Yannakakis, and Y. Zhang. Alembic: Automated model inference for stateful network functions. In *Proceedings of the 16th USENIX Conference on Networked Systems Design and Implementation*, NSDI '19, pages 699–718, USA, 2019. USENIX Association.

[32] NS3 Development Team. NS3 discrete-event network simulator for Internet systems, 2011.

[33] R. Potharaju, N. Jain, and C. Nita-Rotaru. Juggling the jigsaw: Towards automated problem inference from network trouble tickets. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 127–141, Lombard, IL, Apr. 2013. USENIX Association.

[34] B. Schlinker, R. N. Mysore, S. Smith, J. C. Mogul, A. Vahdat, M. Yu, E. Katz-Bassett, and M. Rubin. Condor: Better topologies through declarative design. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 449–463, New York, NY, USA, 2015. Association for Computing Machinery.

[35] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, H. Liu, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, and A. Vahdat. Jupiter rising: A decade of Clos topologies and centralized control in Google's datacenter network. *Commun. ACM*, 59(9):88–97, Aug. 2016.

[36] J. Sommers, H. Kim, and P. Barford. Harpoon: A flow-level traffic generator for router and network tests. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '04/Performance '04, page 392, New York, NY, USA, 2004. Association for Computing Machinery.

[37] D. Turner, K. Levchenko, A. C. Snoeren, and S. Savage. California fault lines: Understanding the causes and impact of network failures. *SIGCOMM Comput. Commun. Rev.*, 40(4):315–326, Aug. 2010.

[38] K.-K. Yap, M. Motiwala, J. Rahe, S. Padgett, M. Holliman, G. Baldus, M. Hines, T. Kim, A. Narayanan, A. Jain, V. Lin, C. Rice, B. Rogan, A. Singh, B. Tanaka, M. Verma, P. Sood, M. Tariq, M. Tierney, D. Trumic, V. Valancius, C. Ying, M. Kallahalla, B. Koley, and A. Vahdat. Taking the edge off with espresso: Scale, reliability and programmability for global internet peering. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, pages 432–445, New York, NY, USA, 2017. Association for Computing Machinery.