KnowOps: Towards an Embedded Knowledge Base for Network Management and Operations

Xu Chen[†] Yun Mao[†] Z. Morley Mao[§] Jacobus Van der Merwe[†]
[†] *AT&T Labs - Research* [§] *University of Michigan - Ann Arbor*

***Abstract*—The domain knowledge required to manage and operate modern communications networks is still largely captured in human-readable documents. In this paper we take the position that an embedded machine readable knowledge base that directly supports network management and operations systems is required. We present a framework for such an approach, called *KnowOps*, and illustrate how it complements and enhances state-of-the-art network management and operation systems.**

I. Introduction

The management and operations of modern networks and network services involve an overwhelming array of operational tasks. For example, they typically include configuring tens of thousands of devices, housed in thousands of physical locations, often distributed across all continents; dealing with (on a typical day) hundreds of planned maintenance activities, tens of mass traffic events, tens of cable cuts and thousands of hardware failures; processing (again on a typical day) billions of measurements and test probes, and millions of alarms, which can result in thousands of actionable tickets and tens of service disruption reports; and, finally, dealing with a major networking event often every other day.

State-of-the-art network management and operation infrastructures that enable operators to deal with the enormous complexity and the scale of these tasks consist, by itself, of complex software systems. Capturing and transferring the collective knowledge base of what it takes to manage and operate the network is a significant part of what these software systems encode. Unfortunately, the capturing and transferring of this knowledge base, mostly happens in a traditional knowledge transfer and software development cycle. I.e., in somewhat simplified form: equipment vendors describe the capabilities of their equipment, how it is configured and what alarms it will generate in prosaic vendor documents, network operator domain experts (i.e., network engineers and network operators) read these vendor documents and use them as input to write network and service design documents for the various operational tasks that operators face. The operator design documents are then in turn used as input to software development teams which encode the functions (and knowledge base) into software systems.

In this paper we argue that, instead of capturing and transferring the knowledge base associated with this process manually via human readable documents, the knowledge base itself should be encoded and captured in a systematic machine readable framework, which can serve as input to the various stages of the design process and be extended as needed by each stage. Ideally this systematic framework should allow domain experts to directly express their designs, without requiring them to become software engineers. In this manner, the knowledge base in effect becomes naturally embedded in the network management and operations system itself.

The outline for the remainder of the paper is as follows. To provide context for our discussion, in §II, we describe the components in a modern end-to-end network management and operations system. In §III we present the *KnowOps* framework based on our philosophy of using embedded knowledge. We argue for the use of a declarative language methodology to form the basis for expressing this systematic knowledge base. We illustrate the benefits of our approach using realistic examples. We present related work in §IV before concluding in §V.

II. State-of-the-art Network Management and Operations

Exactly what is network management and operations? Broadly speaking, network management and operations pertain to all those actions that result in a "healthy" operational network that efficiently support all intended services and prevent unintended uses and abuses. As such it includes: (i) planned maintenance, *e.g.,* to upgrade or introduce new equipment, (ii) emergency repairs, *e.g.,* when a natural or human induced event causes failure or malfunction, (iii) fault management, *e.g.,* to localize and replace faulty equipment, (iv) configuration management, *e.g.,* to enable new functionality or customer features, (v) traffic/performance management, *e.g.,* to deal with traffic growth and dynamic traffic events, (vi) security management, *e.g.,* to handle security incidents like worm outbreaks and DDoS attacks, (vii) network measurement and monitoring, *e.g.,* to detect anomalies, (viii) service management, *e.g.,* the realization and maintenance of new services and service features. While each of these network management/operations functions is distinct, they all pertain to the same "organism", i.e., the network. As such these function are interdependent through the systems and processes that realize their functionality.

Figure 1 depicts a somewhat simplified view of the systems and processes involved in a typical state-of-the-art network operations framework dealing with planned maintenance and fault and performance management [7]. The figure attempts to convey two sets of information. First, on the left, the figure shows the network and the various systems involved in

network operations. Second, the right-hand side summarizes the various inputs to the knowledge base that forms the foundation to the network operations process and how that knowledge base gets applied in the framework. We will now consider the main functional components in more detail.

Network Configuration Management: The functionality and services that a network provides is determined by the collective configuration of all equipment (or all network elements) deployed in the network. As shown in Figure 1, the mechanism whereby network configurations get applied to the system is through a *network configuration management* system, which interacts with the network through a *network interface* abstraction. State-of-the-art network configuration management systems [5] rely on configuration templates (or "configlets"), the parameters of which are populated from a *network inventory* database, before they are pushed to the network to realize network configuration change.

Consider now the knowledge base involved in enabling network configuration management. As depicted in Figure 1, the functionality of network equipments and more specifically the way in which the equipments are to be configured to realize such functionality, is typically described in human readable *vendor configuration manuals*.[1] Service provider networking domain experts (network engineers) interpret these vendor documents and typically experiment in lab environments to produce *provider service design documents*, which capture the specific functionality to provide and the configuration changes required to realize that. Such provider design documents might typically contain configuration templates and explain how these templates are to be parametrized from network inventory databases. Provider service design documents typically serve as input to software system domain experts, who produce *systems design documents* (or more typically, produce system change design documents) from which software teams write code to add the desirable functionality to the network configuration management system. This clearly is a very human labor intensive and potentially error-prone process.

Monitoring and Event Correlation: Once a network service has been successfully deployed and configured, both network and service specific monitoring needs to be deployed to support its continued operation. Figure 1 depicts a simplified view of how this is achieved. First, appropriate measurements and/or monitoring needs to be deployed or enabled as part of a *network instrumentation* layer. This would typically include both passive monitoring, e.g., receiving SNMP traps, as well as active monitoring, e.g., periodic `ping` tests or other service-specific network measurements. Huge volumes of unstructured monitoring data is received in this manner. To facilitate (near) real-time fault and performance management,

[1]Vendors are migrating to systematic machine readable descriptions of their devices, e.g., using XML schemas. We note, however, that this typically provides structure to the syntax of configuration, but does not deal with the semantics of configuration. Further, where this is available, the machine readable specifications deal with network configuration on a per-device basis, as opposed to a network-wide view that is ultimately required.

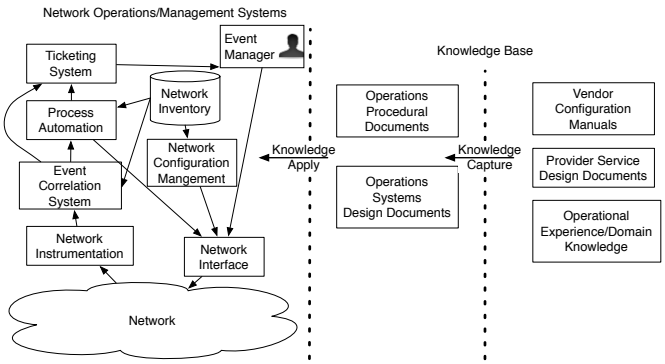

Fig. 1. State-of-the-art Network Management and Operation

an *event correlation system* is deployed to add structure to this data, as shown in Figure 1.[2] State-of-the-art event correlation systems [11] make use of service specific temporal and spatial models to capture service and network dependencies to allow related low level events to be correlated thus reducing the volume of events and more importantly providing more specific ("actionable") information to the higher layers of the network operations framework.

In terms of knowledge capture and transfer to realize network instrumentation and event correlation systems, an analogous, but parallel, process to configuration management is followed. Again vendor documentation informs provider design processes which are captured in provider design documents which serve as input to actual system development. More so than in the case of configuration management, provider *operational experience* and *domain knowledge* come into play as providers need to take a truly holistic network-wide perspective cutting across devices (possibly from different vendors), across network layers and across services. Again this knowledge informs the design process and ultimately gets encoded in software systems.

Event Management and Automation: Returning to the left-hand side of Figure 1, the output of an event correlation system often feeds into a *ticketing system* which keeps track of actionable events being worked by human *event managers* (i.e., operations domain experts). Depending on their skill level, these event managers might rely on their own domain knowledge, or, more typically, follow instructions from knowledge captured in *operations procedural documents*. State-of-the-art operations frameworks allow for automation of many of the more mundane operational tasks through a *process automation* system [7]. A process automation system take the output of an event correlation system as input and allows operators to specify rules to detect well understood conditions and the corresponding action that need to be taken. Rules are typically of the `if <condition> then <action>` type and as shown in Figure 1, the automation system can interact with the network (via the network interface abstraction), so that actions can be of arbitrary complexity. A typical

[2]As part of a comprehensive network management and operation framework, data from the network instrumentation layer would typically also be archived to enable off-line processing and analysis.

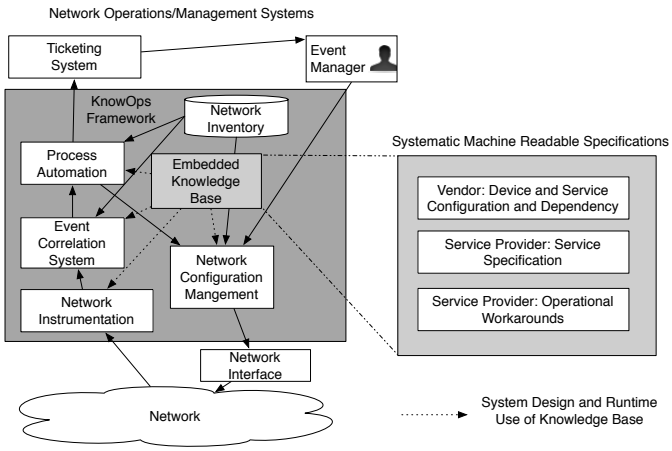

Fig. 2. KnowOps: Network Management and Operation with an Embedded Knowledge Base.

action might interact with the network to automatically collect more information regarding an event, e.g., `show` commands on a router, which could be added to the actionable event information passed to the ticketing system. More sophisticated actions taken by the process automation system might include performing actions described by operations procedural documents (once they have been transformed into machine executable code). Considering how the knowledge base gets applied for this part of the network operations framework, we note that domain knowledge drives the rules and subsequent actions that are encoded in the process automation system.

The description in this section is by necessity incomplete and at a high level. However, it should be sufficient to illustrate our position that in state-of-the-art network operations frameworks, obtaining and capturing domain knowledge happen largely in a separate and adjunct manner to how that knowledge is applied. In cases where domain knowledge is directly encoded into network operations systems, e.g., in the event correlation and process automation systems described above, this happens in an ad-hoc and standalone fashion where in effect each system starts with an "unstructured landscape" and attempts by itself to add structure to that landscape without any systematic support from the operational framework. In essence the knowledge base that holds the system together is still largely based on human readable documents. In the next section we propose a network management and operations framework that systematically captures domain knowledge in an embedded manner so that such domain knowledge can directly inform the design and operation of all operations systems.

III. Embedded Knowledge Base

We envision an environment where different stakeholders and role-players can contribute knowledge, in a systematic, machine-readable manner, so that the knowledge base can be seamlessly integrated into a comprehensive network management and operations framework, and at the same time allow automated reasoning to be performed by network management tools and systems. Figure 2 depicts KnowOps, our proposed framework for an embedded knowledge base for network management and operation. As shown in the figure, the embedded knowledge base in KnowOps directly supports the main network operations systems introduced in Section II, namely configuration management, network instrumentation, event correlation and process automation. Instead of separately deriving templates for managing configurations, writing association rules for correlating events, and composing operational rules for automating processes, we envision these systems to share the same knowledge base.

We expect this sharing to have a positive impact on system development times as system are developed against a common knowledge base. Perhaps more importantly, however, we expect our approach to cut down, and possibly eliminate, the chances of inconsistency across support systems, *e.g.,* a slightly different interpretation of the configuration files may result in drastic difference in the understanding of network functionality.

Below in Section III-A, we first consider how a knowledge base would be established, and the resulting requirements this imposes on such a framework. In Section III-B, we consider how an embedded knowledge base might add value to the configuration management, event correlation and process automation functions. Finally, in Section III-C we consider the more ambitious goal of KnowOps developing into a comprehensive reasoning framework whereby fundamental network properties, *e.g.,* the way a particular protocol behaves, can simply be "plugged into" the framework with operations components automatically adapting their behavior.

A. The KnowOps Knowledge Base Framework

A key question to consider is how an embedded knowledge base would be established. As depicted in Figure 2, our position is that the knowledge base would essentially be contributed by the same role players that do so today. I.e., equipment vendor personnel, network engineers and operations personnel; however, instead of doing that through human readable manuscripts, they would do so through machine readable specifications. This has a number of fundamental implications:

Ease of use: The domain experts who would contribute to this knowledge base are in general not software developers. However, they typically are very logical and systematic in their work. Ideally then, a knowledge framework would be easy to use and provide users with a toolset that can capture their design, without requiring them to become low level code writers.

Comprehensive tools: An ideal knowledge framework would also go beyond simply capturing the knowledge, but instead allow tools to reason about and verify the consistency of the knowledge base.

Extensible and/or transposable: The key to our approach is that different role players extend or add to "the same" knowledge base. This calls for a knowledge framework that is either extensible, or can easily be transposed to different frameworks.

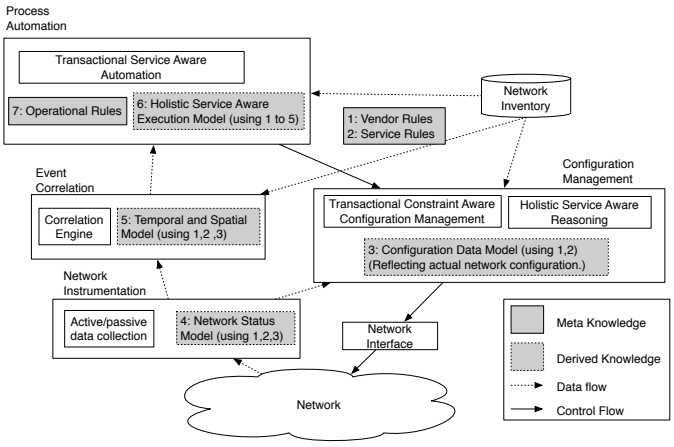

Fig. 3. KnowOps System View

To realize this framework, we are exploring the use of Drools [1], a unified automation framework that combines rule-based and flow-based automation and event correlation. We expect that our approach can be realized through other means; however, the Drools framework is attractive because: (i) It captures the required mechanisms of different network operations in the same framework. (ii) The rules are proven to capture domain knowledge in the realm of automating business logic. (iii) The open-source nature of the software allows us to integrate additional logic to drive different components to better fit network management needs.

B. KnowOps Utility

Figure 3 illustrates in more detail the concept of knowledge sharing across different management aspects in the KnowOps framework. We first differentiate two types of knowledge, meta and derived:

Meta knowledge abstracts how the network should work. It can be provided by vendors (1 in Figure 3), describing device capability, protocol dependencies, *etc.* For example, it might capture the fact that instantiating an instance of a Virtual Private LAN Service (VPLS) between two provider edge routers requires (amongst other things) that the relevant interfaces on both routers be configured and that working instances of the BGP and LSP protocols be operating between the routers. (We explore this example in more detail below.) Meta knowledge can also be specified by service providers (2 in Figure 3), indicating how services should be realized and the associated operational constraints. For example, core routers must form a full BGP mesh. Service providers can also design operational procedures (7 in Figure 3) as part of this meta knowledge, to ensure continuous and satisfactory service delivery. For example, a link must be cost out first by increasing its link weight before it can be shut down.

Derived knowledge is automatically generated to reflect how the network actually operates, based on applying the meta knowledge onto an actual network. Such knowledge is not generated from a single source, but rather refined and reused across different components. First of all, given a list of network devices, their physical status, and the configuration on them, we derive a configuration data model (3 in Figure 3) to capture a network-wide view of the services and functionalities therein. Currently, such a view must be built via mental reasoning or specialized support systems. We develop a network status model (4 in Figure 3) to give structure and hierarchy to a sea of network events that are previously stored in a flat space. We further build a temporal and spatial model (5 in Figure 3) such that network events are correlated together to identify root causes. Note that the current event correlation system depends on a manual specification of causal relationships across events, a process that is unavoidably tedious, likely incomplete, and sometimes incorrect. In KnowOps, such rules are systematically and automatically derived from the network itself. Finally, we establish an execution model (6 in Figure 3) to enhance process automation, *e.g.,* by intelligently scheduling task executions to avoid negative network impact and prioritize high-value services. Such process automation has been mostly done by simple re-execution of stored procedures, without considering network-wide effects.

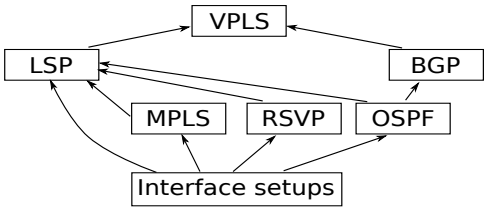

Fig. 4. Simplified VPLS related dependency graph

We now explain in detail how individual management components benefit from such a knowledge base by using VPLS VPNs as an example of a reasonably complex service offered by ISPs. Figure 4 shows a simplified view of the levels of protocols that must be configured to enable a VPLS service that provides a layer-2 connectivity across different customer sites. Each box in the figure represents configuration elements on a distributed set of routers. For example, setting up a VPLS instance requires configuring customer-facing interfaces, as well as establishing iBGP sessions (for control plane signaling) and label switching paths (LSPs, for data plane) between the PE routers. A working LSP in turn depends on configuring the core routers to enable various distributed protocols, including MPLS, RSVP, and OSPF.

Configuration management: The ability to holistically reason about network services is essential. From a bottom-up perspective, we need to reason about the current network inventory, status, and configuration to understand existing network status, e.g., answering questions like *How many VPLS instances are enabled? Can the LA site reach NY site for customer C?* Answering these questions currently requires either human reasoning or tailored support systems. At the same time, we need to estimate the impact of a candidate change to the network, *e.g., Are any VPLS customers impacted if this core link L is shut down?* From a top-down perspective, we must derive a set of low-level operational changes to fulfill a high-level intention, *e.g., What configuration on what devices should be changed to enable a VPLS customer*

connection? What is lacking today is also the support of integrating such reasoning capability with operational logic. For example, instead of blindly executing a sequence of network configuration changes, a more sophisticated method is to anticipate the potential impact and reject operations, like database transactions, to prevent undesired outcomes such as misconfigurations from making into the network.

In our earlier work on configuration management we developed the COOLAID system [2], which is an example configuration management system following a declarative language approach. As such, COOLAID supports an explicit knowledge base specified using a declarative language. Following this approach, if new services are offered or new misconfiguration types are identified, device vendors and service providers can simply provide new rules (as 1 and 2 in Figure 3). One of COOLAID's key features is the ability to perform holistic service-aware reasoning. By using a set of declarative rules, COOLAID is able to derive network-wide services based on low-level configurations, and automate configuration changes from high-level intentions, *e.g.,Configure a VPLS instance spanning PE1 and PE2.* Providing a database-like transaction logic, COOLAID can reject operations that cause misconfigurations and perform automatic role-back.

Event correlation: The ability to efficiently identify root causes of network-impacting failures from a voluminous collection of raw network events is critical for this component. At the network instrumentation layer, we must place various types of sensors to capture the running status of a network. Due to the scale of modern networks and the deployment and run-time cost of the monitoring infrastructure, network designers must make explicit trade-offs between accuracy and cost. At a higher correlation layer, events that are results of the same root cause should be grouped together, such that failure mitigation systems can act properly.

We can use the knowledge resulting from the configuration model in KnowOps to inform and enhance the network instrumentation. First, we can associate the events to a hierarchy of services. Given the relative importance of different services, we can intelligently determine the placement, granularity, delivery mechanisms, *etc.* for the network sensors, *e.g.,* monitor a small subset of routers to capture the top-10 VPLS customers' traffic. More importantly, such derived knowledge is directly based on the actual dynamic network setup, and thus the instrumentation layer can be reconfigured accordingly.

Current event correlation depends on the manual or statistically inferred specification of causal relationships. For example, a broken BGP session event and a disrupted VPLS connection event that happened within a small time window are correlated by applying a temporal rule. However, not all BGP session downs will impact VPLS connections, so additional spatial rules must be defined, *e.g.,* the downed BGP session must be connecting the two PEs of the VPLS connection. In KnowOps, such causal relationship is accurately captured by the configuration model and virtually comes for free by reusing it from the configuration management component. For example, COOLAID calculates the IGP path between two PE routers, thus gathers a set of links that might impact the higher layer BGP session.

A key challenge of event correlation systems is data loss. For example, we see a broken VPLS service, but may not have any information regarding the underlying BGP and LSP, which VPLS depends on, because of loss of events, delayed delivery, *etc.* In existing systems, a common approach is to simply ignore certain messages, which could lead to incorrect understanding of the network. In KnowOps, since all the dependencies are known in the configuration model, we can easily identify the immediate dependent services and actively retrieve the missing information from the network without tolerating missing data.

Process automation: Various level of automation exists in the current service provider environments. However, because modern networks are shared in nature, any changes to a production network have the potential of negatively impacting existing services. Existing automation support, such as script executions or more sophisticated automation engines, which mostly focuses on individual and specific tasks, can significantly cut down manual involvement, but still leaves the operators' expertise and manual involvement to ensure network-wide wellness. Indeed, it is unreasonable to require procedure (operation rule) designers to be aware of and encode the handling of all the possible outcome and impact on the network. For example, a procedure to automate a link maintenance might be fine to execute in most cases because of the resilience of an over-provisioned backbone, but such automation should be stopped during a DDoS attack or a peak hour due to a lack of available bandwidth. The blind automation of any procedures without staying cognizant of the network status will unavoidably cause network disruption.

By reusing the knowledge from the configuration model, we can reason about the impact of and intelligently schedule the network operations as the automation procedure progresses. As a result, the network designers can compose operational rules in a familiar task-centric fashion. Similarly, with the knowledge about the actual services running in the network, such process automation engine has the visibility into the importance of individual tickets it receives, *e.g.,* some link down events can be quickly recovered by IGP, while others may cause network partitioning. By prioritizing the handling of the events that have largest impact on the network, we can improve network reliability and deliver higher service guarantee to customers.

C. KnowOps as a Reasoning Framework

We observe that a significant part of the (sometimes implicit) reasoning happening in the functions considered above relate to modeling how the network behaves, or should behave. Returning to our running example, configuring a VPLS instance in effect reasons about the protocol dependencies and how they need to be configured to realize the configuration goal. Event correlation reasons about the spatial and temporal dependencies of events based on the expected behavior of

the network (or indeed sometimes looking for what is not expected). This level of reasoning depends on a knowledge base that captures service and protocol dependencies as we have described above.

A more sophisticated level of reasoning might be needed if understanding the *dynamic* nature of protocols are required. For example, a traffic engineering system would require both an understanding of the topology of the network and how it is configured, but also need to understand more detail about how specific protocol decisions, *e.g.,* OSPF tie-breaking, is implemented by the particular vendor implementation. Today this kind of functionality is achieved by standalone systems that reverse engineer these low level protocol details [10]. We envision that the KnowOps framework would readily support this kind of functionality provided that the knowledge base accurately reflect the actual protocol implementation.

A more ambitious solution would be to develop a generic mechanism that reasons and acts based on "pluggable protocol knowledge". For example, a generic monitoring and fault diagnosis framework where we can just plug in the knowledge representing the details of the underlying protocols. In the first instance such an approach would again rely on the knowledge base accurately reflecting the actual protocol implementations. In a more extreme version of this, the network implementation itself would be driven by a declarative expression of the networking protocols. Such an approach has been advocated [8], although without an understanding of how that might impact network management and operations.

IV. Related work

Managing computer networks or distributed systems with various forms of knowledge representation is a topic that continuously receives attention. Most notably, Clark *et al.* proposed the concept of knowledge plane [4], a distributed cognitive system permeating the network. The 4D project includes a decision plane, which builds a network-wide view and issues control over the network elements [6], [12]. In essence, KnowOps is similar to these approaches, but with a more modest focus of targeting existing network management systems in a single ISP setup.

PACMAN [3] represents a system that uses a petri-net model to capture network operational workflow logic and further encode network-wide reasoning into the execution of management tasks. COOLAID [2] proposes to use a declarative language to capture domain knowledge from both device vendors and service providers, such that the resulting rules can be applied onto a database-like abstraction of an entire network to automate various network operations. KnowOps unifies these systems in the same framework, and further extends using the same knowledge base to more general network operations, such as fault and performance management.

Many past projects have focused on designing effective troubleshooting, root cause analysis, and diagnosis support for large IP networks. Our work is complementary to these systems in the objective of providing the event correlation utility. For instance, The G-RCA system [11] is a generic root cause analysis platform for service quality management system based on a comprehensive service dependency model and allows customization by operators using a rule specification language. Within the same problem space, the NICE system [9] is an infrastructure to troubleshoot chronic network conditions using statistical correlation across multiple data sources. KnowOps provides for a more formal and systematic knowledge base to replace the somewhat ad-hoc approaches adopted by these systems.

V. Conclusion

In this paper we take the position that an embedded, machine readable knowledge base is essential to comprehensively tie together the various systems that make up an end-to-end network management and operations infrastructure. We presented such a framework and argued that a declarative language approach presents attractive properties to form the basis for such an embedded knowledge base. We showed, by way of considering the functions involved with managing a real world service, how such an embedded approach provide benefits over current state-of-the-art network management and operations systems.

References

[1] Drools: Business logic integration platform. http://www.jboss.org/drools.

[2] Xu Chen, Yun Mao, Z. Morley Mao, and Jacobus van der Merwe. Declarative Configuration Management for Complex and Dynamic Networks. In *Proceedings of ACM CoNEXT*, 2010.

[3] Xu Chen, Z. Morley Mao, and Jacobus Van der Merwe. PACMAN: a Platform for Automated and Controlled network operations and configuration MANagement. In *Proceedings of ACM CoNEXT*, 2009.

[4] David D. Clark, Craig Partridge, J. Christopher Ramming, and John T. Wroclawski. A knowledge plane for the internet. In *Proceedings of ACM SIGCOMM*, 2003.

[5] William Enck, Patrick McDaniel, Subhabrata Sen, Panagiotis Sebos, Sylke Spoerel, Albert Greenberg, Sanjay Rao, and William Aiello. Configuration management at massive scale: system design and experience. In *Proceedings of USENIX ATC*, 2007.

[6] Albert Greenberg, Gisli Hjalmtysson, David A. Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, Jibin Zhan, and Hui Zhang. A Clean Slate 4D Approach to Network Control and Management . In *Proceedings of ACM SIGCOMM CCR*, 2005.

[7] Charles R. Kalmanek, Sudp Misra, and Y. Richard Yang, editors. *Guide to Reliable Internet Service and Applications*, chapter Network Management: Fault Management, Performance Management and Planned Maintenance. Springer, 2010.

[8] Boon Thau Loo, Joseph M. Hellerstein, Ion Stoica, and Raghu Ramakrishnan. Declarative routing: Extensible routing with declarative queries. In *Proceedings of ACM SIGCOMM*, 2005.

[9] Ajay Mahimkar, Jennifer Yates, Yin Zhang, Aman Shaikh, Jia Wang, Zihui Ge, and Cheng Ee. Troubleshooting Chronic Conditions in Large IP Networks. In *Proceedings of ACM CoNEXT*, 2009.

[10] Aman Shaikh and Albert Greenberg. OSPF Monitoring: Architecture, Design and Deployment Experience. In *Proceedings of NSDI*, 2004.

[11] He Yan, Lee Breslau, Zihui Ge, Dan Massey, Dan Pei, and Jennifer Yates. G-RCA: A Generic Root Cause Analysis Platform for Service Quality Management in Large IP Networks. In *Proceedings of ACM CoNEXT*, 2010.

[12] Hong Yan, David A. Maltz, T. S. Eugene Ng, Hemant Gogineni, Hui Zhang, and Zheng Cai. Tesseract: A 4D Network Control Plane. In *Proceedings of NSDI*, 2007.