Towards Self-Adaptive Network Management for a Recursive Network Architecture

Jason Barron*, Micheal Crotty*, Ehsan Elahi*, Roberto Riggio†, Diego R. Lopez‡
and Miguel Ponce de Leon*
*Telecommunications Software & Systems Group
Waterford Institute of Technology, Waterford, Ireland X91 P20H
Email: jbarron, mcrotty, eelahi, mpdl@tssg.org
†CREATE-NET, Italy
Email: roberto.riggio@create-net.org
‡Telefonica I+D, Zurbaran 12, 28010 Madrid, Spain
Email:diego.r.lopez@telefonica.com

***Abstract*—Traditionally, network management tasks manually performed by system administrators include monitoring alarms based on collected statistics across many heterogeneous systems, correlating these alarms to identify potential problems or changes to management policies and responding by performing system re-configurations to ensure optimal performance of network services. System administrators have a narrow focus of factors impacting network service provisioning and performance due to the heterogeneity and scale of generated underlying network events. However, self-adaption principles are conceptual approaches for autonomously managing such complex distributed systems. Network management systems that harness such principles can dynamically and autonomously optimise the operation of network services, responding quickly to changes in user requirements and underlying network conditions. In this paper, we present a novel self-adaptive network management framework that takes advantage of a recursive network architecture for a simpler and more comprehensive application of ontologies, semantic web rules and machine learning to form the basis of an autonomic network management system that can automatically adjust network configuration parameters to provide more optimal QoS management of network services. We demonstrate the applicability of the approach using a content distribution network (CDN) operating over such a recursive network architecture.**

I. Introduction

In traditional network management approaches, network re-configurations are manually performed by administrators based on collected statistics, alarms, or even by changing management policies. System administrators have a narrow view of correlated network events, as it is difficult to understand and leverage the large amount of heterogeneous data generated and collected from a multitude of network systems and devices at runtime. This problem worsens as the network data-set increases with more complex event correlation required.

Self-adaption principles known as the self-* properties (i.e. self-configuration, self-healing, self-optimisation and self-protection) [1] are conceptual approaches for autonomously managing complex distributed systems. Network management approaches that employ such principles can dynamically change network service behaviour without manual system administrator intervention, responding more quickly to changes in user requirements or the underlying networking environment and with less risk of introducing network configuration errors. In this paper we investigate how self-adaption principles can be better applied to network management tasks when a recursive network architecture is used, facilitating the optimisation of network service operations at runtime.

We propose a novel self-adaptive network management approach capable of autonomously managing network services by rigorous analysis of low-level network data to make high-level decisions in order to optimise network service performance. The goal is to develop a solution that provides a higher, more intelligent management of network services and applications by improving operational efficiencies and facilitating the requirements of an autonomous network management system for a recursive network architecture. The self-adaptive framework correlates low-level network events and alarms to recognise pertinent network event patterns that are analysed using Machine Learning (ML) algorithms to yield insights and performance predictions, responding to them autonomously by modifying underlying networking configurations as required. This allows the network management system to adapt to and actively learn from changes in network service characteristics and demands that can occur where demand may vary over the lifetime of the network service.

The approach exploits leading research in the areas of data gathering, complex event correlation, machine learning, data analytics and self-adaptive network management principles to improve network service performance; beneficial for both end-users in terms of Quality of Experience (QoE), and network service operators by lowering capex and opex costs through improved operational efficiencies for network service providers. This allows network service operators scale network services in volume and complexity while still maintaining optimal network manageability achieving objectives such as reliability and efficiency.

The outline of this paper is as follows: §II provides context and motivation for the work; §III outlines the self-adaption framework. §IV outlines a typical use case for self-adaptive network management of a content distribution network. §V provides an implementation of the self-adaptive network management approach. §VI outlines related work in the area; Finally, §VII summarises the paper and outlines directions for further work.

II. Context / Motivation

In the last few years, interest has grown in Future Internet architectures. This interest is mainly driven by the pragmatic concerns of large scale ISPs and data center deployments, cloud providers and businesses that want a more adaptable, configurable, flexible, resilient network on which to build network services for end-users. These goals have been addressed by applying different, and orthogonal, enabling technologies that have grown during recent years, such as network virtualisation [2] (deploy separate logical networks on a common networking infrastructure), Software Defined Networking [3] (programmable and logically centralized control over packet flows) and Network Function Virtualisation [4] (decouple network capacity and functionality by applying cloud principles). However, these approaches are stand-alone point solutions that address a particular aspect of the problem space. Whereas, a more encompassing approach is the recursive internetwork architecture (RINA) [5], an emerging clean-slate programmable networking approach that aims to support high scalability, multi-homing, built-in security, seamless access to real-time information and operation in dynamic environments. RINA takes as a starting point the basic premise that networking is inter-process communication (IPC) and only IPC [6], and the network is modelled as the interconnection of applications over different scopes, known as Distributed Application Facilities (DAFs). RINA considers one single layer, recursively repeated as needed, implementing specialized DAFs (i.e. IPC processes on each system work together to form a Distributed IPC Facility (DIF)), and two protocols at each layer: one for data transfer, with a consistent QoS model [5], and another for application (layer) management. RINA uses a Resource Information Base (RIB) for the logical representation of information held by the IPC Process (IPCP) for the operation of the DIF. RINA conceives a DIF Management System (DMS) as a centralized tool to perform management tasks over the systems of the network capable of making complex configuration changes affecting multiple layers at once and optimizing the performance of a set of layers working together. The DMS follows a manager-agent (MA) model for its network management tasks using two protocols, the Common Application Connection Establishment Phase (CACEP) allowing application processes to establish an application connection and the Common Distributed Application Protocol (CDAP) enabling distributed applications to communicate at an object level rather than using serialisation to assist the DMS runtime operations.

III. Semantic Network Management Framework

This section describes the individual self-adaptive functions within the overall self-adaption framework, outlined in Figure 1, that includes event monitoring and correlation for identifying pertinent network events, machine learning for both intra- and inter layer cognitive network management and model-driven development techniques for DIF layer instantiation and re-configuration.

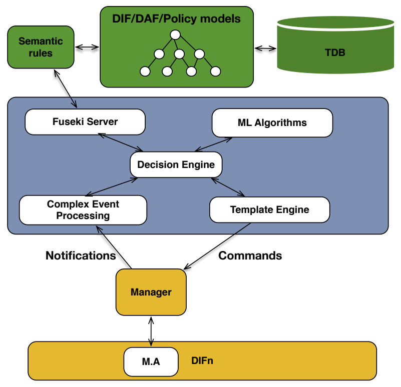

Fig. 1. Semantic Network Management Framework

Ontology-driven network management [7] using ontological models offer powerful modelling constructs to represent the structural and behavioural aspects of managed network entities and more importantly enables semantic reasoning over the relationships that exist between those managed network entities for further analysis [8]. Domain adaptation methods enable the application of machine learning techniques to correlated network events, where a large number of heterogeneous events requires a higher level of abstraction so that automatic classifiers can effectively make more reliable network service predictions. The automatic self-adaption mechanisms enable self-control (i.e. self-FCAPS [9]) of underlying network operations, functions and state minimising expensive, manual system administrator intervention.

The semantic network framework correlates generated alarms with their associated DIF and calculates the effective network management impact caused by the alarm and specifically application impact. For example, analysis on the actual usage of network resources as opposed to the declared expected usage or if there are historical basis to these alarms (e.g. greater than 20% under utilisation, greater than 10% admission failures, etc.). This allows network service performance tuning, where the allocated resources are optimised to the aggregate needs, increasing resources to over-loaded network services and reducing resources to under-utilised ones. This involves adjusting the DIF configurations and in particular the QoS cubes to a more optimised form.

The framework considers an initial network configuration and re-configurations of the network over the lifetime of network service provisioning for continuous re-configuration updates to provide optimal network service performance. Management policies provide the overall system goals to actively guide the machine learning algorithms for improved decision-making. The framework provides a convenient method to instantiate and destroy DIFs as required and support re-configuration of DIFs programmable behaviours and in particular its QoS cube parameters according to changes in the network service requirements. This implies the discovery, inventory and re-configuration of programmable behaviours within a DIF from the DMS, as well as ensuring consistency on the programmable behaviours across a group of associated DIFs.

A. *Event monitoring & correlation function*

The recursive nature of RINA facilitates a much better aggregation of events and correlation of alarms, with a consistent protocol for data transfer differentiated by means of policies at each layer. This function identifies specific event triggers from deployed DIF configurations and ensures event collection from network systems and devices that involves pre-processing and correlating events to classify the events generated and identify the most important and irregular events for submission to the machine learning function while filtering routine and regular events. This is an important step in the development of a scalable self-adaptive network management solution as it dramatically reduces the scale of events required for processing by the machine learning algorithms.

B. *Intra- & Inter- layer cognitive network management function*

The availability of a single management protocol and consistent object model for the RIBs, supports a simpler and more powerful application of the ontologies and machine learning models. This function applies machine learning algorithms from statistical learning theory to balance desired changes in network configuration with protecting existing configurations.

Specifically, this function analyses changes in network operations (state, configuration, functions) as a result of changes in the underlying network conditions, network service or user context and uses either supervised or unsupervised machine learning algorithms to facilitate domain adaptation by developing a system of service demand prediction and provisioning which allows the network to resize and resource itself to serve predicted demand according to various parameters such as location, time and/or network service demand from specific users or user groups.

This function facilitates addressing network resilience issues, identifying network errors, faults or conditions such as congestion at both a network wide and a local level and automatically prompts mitigating actions to minimise the overall impact on network service provision. This is achieved while optimising performance, use of available underlying network resources and minimising overall network costs for network service providers.

C. *DIF Layer Instantiation/re-configuration function*

Initial DIF layer configuration and re-configuration requires an execution environment, i.e. DMS management policies need to be translated into specific DIF configuration policies and deployed into the appropriate IPC Processes (IPCPs) within each DIF. In particular, this function performs automated instantiation/re-configuration of DIFs through the use of DIF templates by automatically configuring the variable DIF parameters such as name, scope, addressing schema and in the case of performance, QoS cube parameters.

Another feature of this function is to check the consistency of the system state after instantiation, destruction or re-configuration of DIF layers has been performed both within each DIF and between a group of associated DIFs as it is imperative to check that these DIFs have been transitioned to a consistent state. If not, a rollback of the modified DIFs will need to be performed and notification sent to a system administrator to take appropriate remedial action that may include invoking policy analysis processes. This is an area for future work.

IV. Use Case

We provide an automated network service provisioning use case based on a content distribution network which is one of the key services provided by of data centers as it provides the capability of acquiring and releasing underlying networking resources on-demand as required. However, data center operators find it non-trivial to allocate and de-allocate resources dynamically to accommodate network service requirements to satisfy its Service Level Objectives (SLOs) and its Service Level Agreements (SLAs) while minimizing operational costs.

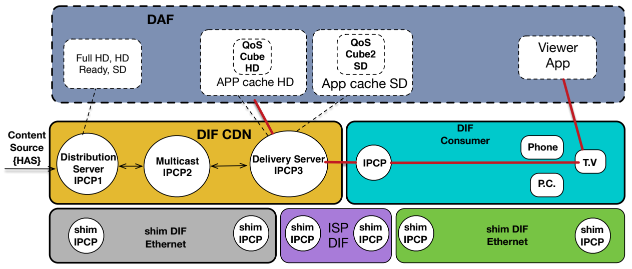

Fig. 2. Use Case Scenario

Our example use case, depicted in Figure 2 is based on a user that started watching a football game on their mobile phone in Standard Definition (SD) quality. However, half-way through the game the user decides to watch the remainder of the game on their T.V. in High Definition (HD) quality. Unfortunately, the current QoS cube provided by the underlying network flow currently only supports a SD quality QoS rate. This means the user's device initiates a flow request to the "App cache HD" application process on the Delivery Server (IPCP3) for a higher rate QoS Cube (HD). There is currently no QoS cube deployed to support HD quality video in the DIF which results in an *allocateNotifyPolicy* firing, and generating an event for the Management Agent. The Management Agent has been pre-configured to report these faults to the DMS.

Within the DMS there are a number of management strategies in operation. One of them is to optimise DIFs such that, unused QoS cubes (daily) are removed from the active DIF configuration. In this case, the QoS cube supporting HD has been garbage collected as there have been no flows in the last two days. A second set of management strategies (for HD) are triggered, and examine the currently available DIFs to see which is the most appropriate DIF that contains the QoS cube supporting HD, from the flow allocation request. The DMS sends a CDAP create operation to the Management Agent to create a QoSCube in the CDN DIF and the consumer DIF. It is assumed that the *allocateRetryPolicy* in the Consumer DIF will retry the flow allocation four times (at 500, 1000, 2000 and 3000 ms). The following list details the steps that are performed:

- Management agent adds a new QoS cube to the DIF specification.
- DIF allocator retries the flow allocation and discovers the DIF Consumer supports it.
- Flow allocator then notifies the application process (in this case the App cache) to accept or reject the flow allocation request.
- Assuming a positive response, the flow allocator reserves required resources, and responds to the TV application.
- A new flow (HD) is instantiated as the response is acted upon.
- Streaming content from old flow (SD to the mobile device) is now streamed to the new flow (HD to the TV).
- A old flow is discarded as the consumer closes the application.

In the above scenario, event correlation is trivial as there is a single event triggering the action. In a more realistic environment, QoS degradation is a possibility leading to multiple QoS Violation events being generated (from each node concerned). Additionally, an application instance may have multiple flows active concurrently, more advanced correlation algorithms are needed to isolate the affected application instance, as multiple instances of the viewing application could be in use on the same network segment.

V. Implementation

Our prototype implementation includes the creation of ontology models used to represent both the RINA network within the data center and the network services running over the RINA network. Jena [10] is an api implemented in the Java programming language used for manipulating ontology models. Some components of the test-bed were implemented in the Java programming language using the Jena API. In order to implement our prototype we used the RINA source code that is available from the GitHub repository [11].

The autonomic control loop used in the autonomic management system is based on control loop [12] that refers to a decision cycle originally designed by John Boyd, a military strategist, and includes the sub-processes of Observe, Orient, Decide and Act (OODA). In our system the sub-processes perform the following tasks:

- **Observe (correlate events)** - Observe network events from systems/devices to identify pertinent events as input to the knowledge model (i.e. pattern matching). Due to the sheer volume of events generated from systems/devices, events need to be aggregated/correlated.
- **Orient (update/query model)** - Sparul is used to update and query the DIF/DAF models to accurately reflect the current runtime state of the system.
- **Decide (ML algorithms over model)** - Machine learning algorithms are run over identified subsets of the knowledge model to make predictions for more optimal service provisioning.
- **Act (create/update configurations)** - DIF templates are populated with QoS cube parameters to either create new DIF configurations or modify existing ones as required (focusing on QoS cube).

The network (DIF/DAF) and policy models were created in the web ontology language (OWL) [13] using Protege [14], a tool for creating and editing ontologies where a network model and a policy model have been defined for both DIF/DAF and management policy models. The Resource Description Framework (RDF) [15] is a language standard for representing information about resources on the web. An RDF triple contains three elements, a subject $\{s\}$, a predicate $\{p\}$, and an object $\{o\}$, these elements are used to form a tuple $\{s, p, o\}$. The tuple represents a property that holds between the subject and object. The property can be a data type or object property. Data type properties refer to the attributes that compose a concept. Object properties refer to the relationship between concepts. Semantic queries are executed over the properties of concepts specified in a knowledge model and are defined over sets of triples to form a basic graph pattern. The semantic queries act as filter returning only a subset of the complete network and policy models. SPARUL [16], a semantic language was used to query and update the domain and policy knowledge bases. Fuseki [17] was used to load the required domain and policy ontologies, issue semantic queries over the loaded ontological knowledge bases and store the results in a data structure.

RabbitMQ [18] an open-source messaging broker based on the Advanced Message Queuing Protocol (AMQP) [19] is used to aggregate and correlate the large number of network events by taking a similar approach as that outlined in [20], so that only the most pertinent network events are used to update the knowledge models and consequently used as input to the machine learning algorithm. Weka [21] is a machine learning software tool written in Java that implements many machine learning algorithms such as naive bayes, bayesian networks and decision tree learners, etc. for performing data analysis and predictive modelling tasks. Weka supports a number of essential data mining functions such as data pre-processing, clustering, classification and regression. In our approach we

TABLE I
PRIORITISATION TRAINING INFORMATION

Priority	1	2	3	4
Delay	100ms	250ms	600ms	2s
Jitter	20ms	45ms	55ms	65ms
Loss	10^{-5}	0.5%	1.5%	4%
Application	VoIP	HD-video	SD-video	Best effort

opted to use Weka's naive bayes algorithm, a well known supervised learning algorithm whose classification approach is based on probabilistic knowledge. Naive bayes is used to classify flow requests to an associated priority as shown in Table I.

Thus each new flow request receives a priority which assists in the decision making process. The premise being that within this consumer DIF, higher priority flow requests (and associated QoS cube) should be accommodated even at the expense of existing lower priority ones. A more complex classifier could also take into account the device initiating the flow and with suitable RINA authentication policies applied on the DIF, the person using the device. This will be a direction for future work. Based on the results of this orientation step, a decision can be made to allow the new flow (and support the associated QoS class) within the current DIF according to high-level management policies that dictate the overall network service behaviour.

Assuming the flow allocation is to be allowed, a simple test is made in the form of a SPARUL query for the existence of an appropriate "priority" QoS cube in the DIF. The result triggers the current DIF configuration to be either maintained as is or modified to support a new QoS class. We used the String Template [22] engine to create pre-defined DIF templates with place-holders for the various QoS Cubes for DIFs. If a DIF configuration change is required, the DIF QoS place-holders are populated with the appropriate QoS parameters for that network service according to the goals of the management policy and deployed onto the devices.

VI. RELATED WORK

A. Protocols and platforms

A large number of protocols exists to support network management applications. Common protocols include SNMP [23], ICMP [24], and NetConf [25]. A database with more than 400 network measurement and management tools is maintained by the MOME project [26]. Most of such tools are designed around the characteristics, and thus the limitation, of the TCP/IP which in time often results in a collection of ad-hoc monitoring and management tools aimed at addressing the requirements of a particular part of the networking stack. Example includes tools for wireless network troubleshooting like Kismet or traffic analysis tools like Bro. In this context, when a problem is recognized, the centralized/distributed monitoring processes usually escalate the event to the management plane where complex root-cause analysis algorithm need to be implemented in order to correlate the event happening at various layers of the networking stack. Conversely in this work we leverage on an recursive network architecture where each layer is architecturally the same, which in time allows to reason about the current state of the network.

B. Autonomic Network Management

In [27], a conceptual architecture for autonomic computing is sketched. The authors analyse the motivation behind the quest for autonomic computing and focus on the control loops introduced by the self-management routines. Several control disciplines are identified (e.g. self-configuring, self-healing, etc) together with the need for an high level orchestrations among them in order to control the mutual interaction of control loops, that could otherwise lead to unpredicted and possibly disruptive effects. In the architecture envisioned in [27], control loops leverage a common knowledge of the system. Along this line, in [28] the authors summarize this need in the Knowledge Plane concept. In the envisioned scenario, the Knowledge Plane is supposed to collect information about the network status as well as about services constraints and polices. A comparison with the Internet is due in this case. The current Internet is the offspring of a simple idea: building a transparent core network and move all the complexity to the edges. This approach led to a situation where the network core is not aware of its expected behaviour and the end-user applications cannot tell whats happening in the network that appears to them as a black box. According to the authors of [28], the use of a Knowledge Plane has two main targets. On one hand it will make a network able to describe itself, and as result capable of supporting self-configuring and self-healing operations. On the other hand it should give the applications the capability to reason about the operating environment. Unfortunately the achievement of such goals goes through an architecture design following the philosophy of saying as little as possible about the network state, while, on the other hand, RINA aims at describing as much as possible about the network, paving the way to actual Knowledge Plane implementation.

C. Software Defined Networking

Software-defined networking has recently emerged as a supposedly new way of re-factoring a network control plane. By moving all control intelligence into a logically centralized controller, SDN aims at simplifying the life of network developers that can now exploit a clean and consistent interface to the networking fabric. In this context OpenFlow [8] has emerged as the de-facto standard (especially in the data-centres) environment for SDNs. Unfortunately, the expressiveness of the OpenFlow protocol is severely limited, and it has been argued that programming current networks using OpenFlow is equivalent to programming applications in assembler, i.e. the interface is too lowlevel and exposes the programmers with too many implementation details. As a result, we have witnessed a plethora of efforts in recent years aimed at providing developers with higher level interfaces to their SDN [29], [30], [31], [32], [33], [34], [35]. These approaches however still provide point solutions and patches to an highly heterogeneous architecture, while the approach presented in this work builds upon a clean-slate design (RINA) which clearly separate policies from mechanism putting the former in the hands of the network developers.

VII. Conclusions

In this paper we have demonstrated an approach to autonomic network management, able to support optimal network service management in the current environments in which traffic evolves towards increasing volume and complexity. To achieve this aim, we investigated and implemented a self-adaptive network management framework to autonomously control the provision of network services running over RINA-based networks, taking advantage of the nature of this recursive architecture to simplify the application and intensify the suitability of semantic modelling and machine learning techniques. Given that there will be a large number of network events/alarms triggered in a multi-layer management framework, future work will investigate how to efficiently monitor and analyse these numerous events as the system scales. Future work will also investigate mechanisms for verification of DIF configurations after modifications have been performed to avoid introducing inconsistencies into the system. We also plan to expand the use case to include scenarios where network services cannot be readily replicated and cached such as online multi-player gaming scenarios.

Acknowledgement

This research is funded by the European Commission through the PRISTINE project (Grant 619305), part of the Future Networks objective of the Seventh Framework Programme (FP7).

References

[1] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[2] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862–876, 2010.

[3] N. McKeown, "Software-defined networking," *INFOCOM keynote talk*, vol. 17, no. 2, pp. 30–32, 2009.

[4] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *Communications Surveys & Tutorials, IEEE*, 2015.

[5] J. Day, *Patterns in network architecture: a return to fundamentals*. Pearson Education, 2007.

[6] J. Day, I. Matta, and K. Mattar, "Networking is ipc: a guiding principle to a better internet," in *Proceedings of the 2008 ACM CoNEXT Conference*. ACM, 2008, p. 67.

[7] J. Strassner, J. N. De Souza, S. Van der Meer, S. Davy, K. Barrett, D. Raymer, and S. Samudrala, "The design of a new policy model to support ontology-driven reasoning for autonomic networking," *Journal of Network and Systems Management*, vol. 17, no. 1-2, pp. 5–32, 2009.

[8] J. Barron and S. Davy, "Semantic web technologies to aid dominance detection for access control policies," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 2013, pp. 780–783.

[9] H.-G. Hegering, S. Abeck, and B. Neumair, *Integrated management of networked systems: concepts, architectures, and their operational application*. Morgan Kaufmann, 1999.

[10] B. McBride, "Jena: A semantic web toolkit," *Internet Computing, IEEE*, vol. 6, no. 6, pp. 55–59, 2002.

[11] I. Stack, "Rina implementation," https://github.com/IRATI/stack, 2015, accessed: 2015-11-10.

[12] J. R. Boyd, "The essence of winning and losing," *Unpublished lecture notes*, 1996.

[13] B. Motik, P. Patel-Schneider, B. Parsia, C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler *et al.*, "Owl 2 web ontology language: Structural specification and functional-style syntax," *W3C Recommendation*, vol. 27, 2009.

[14] J. Gennari, M. Musen, R. Fergerson, W. Grosso, M. Crubézy, H. Eriksson, N. Noy, and S. Tu, "The evolution of protégé: an environment for knowledge-based systems development," *International Journal of Human-Computer Studies*, vol. 58, no. 1, pp. 89–123, 2003.

[15] O. Lassila, R. Swick *et al.*, "Resource description framework (rdf) model and syntax specification," 1998.

[16] A. Seaborne, G. Manjunath, C. Bizer, J. Breslin, S. Das, I. Davis, S. Harris, K. Idehen, O. Corby, K. Kjernsmo *et al.*, "Sparql/update: A language for updating rdf graphs," *W3C Member Submission*, vol. 15, 2008.

[17] Apache, "Fuseki: serving RDF data over HTTP," https://jena.apache.org/documentation/serving_data/, accessed: 2015-11-30.

[18] "RabbitMQ, author=Pivatol, howpublished = http://www.rabbitmq.com/, note = Accessed: 2015-10-23, year=2015."

[19] OASIS, "Advanced message queuing protocol," https://www.amqp.org/, 2015, accessed: 2015-10-23.

[20] S. Dawar, S. van der Meer, E. Fallon, J. Keeney, and T. Bennett, "Building a scalable event processing system with messaging and policies–test and evaluation of rabbitmq and drools expert," 2013.

[21] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[22] T. Parr *et al.*, "Stringtemplate template engine," http://www.ietf.org/rfc/rfc1157.txt, 2004, accessed: 2015-11-04.

[23] J. Case, M. Fedor, M. Schoffstall, and D. J., "IETF RFC 1157: A simple network management protocol," http://www.ietf.org/rfc/rfc1157.txt, May 1990, accessed: 2015-11-30.

[24] J. Postel, "IETF RFC 0792: Internet control message protocol," http://www.ietf.org/rfc/rfc0792.txt, September 1981, accessed: 2015-12-01.

[25] TERENA, "MOME project: Cluster of european project aimed at monitoring and measurement," http://www.ist-mome.org/, March 2006, accessed: 2015-12-01.

[26] R. Enns, "IETF RFC 4741: Netconf configuration protocol," http://tools.ietf.org/html/rfc4741, December 2006, accessed: 2015-11-30.

[27] P. A. Computing, "Roadmap to self managing technology," *An IBM Journal Paper*, 2006.

[28] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A knowledge plane for the internet," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2003, pp. 3–10.

[29] T. L. Hinrichs, N. S. Gude, M. Casado, J. C. Mitchell, and S. Shenker, "Practical declarative network management," in *Proceedings of the 1st ACM workshop on Research on enterprise networking*. ACM, 2009, pp. 1–10.

[30] C. Monsanto, J. Reich, N. Foster, J. Rexford, D. Walker *et al.*, "Composing software defined networks." in *NSDI*, 2013, pp. 1–13.

[31] A. Voellmy and P. Hudak, "Nettle: Taking the sting out of programming network routers," in *Practical Aspects of Declarative Languages*. Springer, 2011, pp. 235–249.

[32] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: A network programming language," in *ACM SIGPLAN Notices*, vol. 46, no. 9. ACM, 2011, pp. 279–291.

[33] A. Voellmy, H. Kim, and N. Feamster, "Procera: a language for high-level reactive network control," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 43–48.

[34] A. K. Nayak, A. Reimers, N. Feamster, and R. Clark, "Resonance: dynamic access control for enterprise networks," in *Proceedings of the 1st ACM workshop on Research on enterprise networking*. ACM, 2009, pp. 11–18.

[35] C. Monsanto, N. Foster, R. Harrison, and D. Walker, "A compiler and run-time system for network programming languages," *ACM SIGPLAN Notices*, vol. 47, no. 1, pp. 217–230, 2012.