An Efficient Model for Object-Oriented Programming in Software Defined Networks

Eisa A. Aleisa
College of Computer and Information Sciences, Al Imam Mohammad Ibn Saud Islamic University (IMSIU),
Riyadh, Kingdom of Saudi Arabia

Summary
Software-Defined Networks (SDNs) is among the most recent advances of networks. SDNs use a controller device to manage the network switches. The management action includes adding or removing packet-management rules on action tables of switches. This paper introduces a high-level object-oriented network programming language, called ObjNet, to enable coding efficient, yet simple, procedures run by controller to control switches. ObjNet is object-oriented, clearly-organized, and expressive. The paper also presents a novel static semantics to ObjNet.

Key words:
Software-Defined Networks (SDNs), network programming languages, controller-switch architecture, static semantics, syntax, ObjNet

1. Introduction

A network is a set of devices linked to swap data. Switches, routers, and firewalls are among these devices. Switches do the job of forwarding packets according to MAC addresses. Routers direct packets using IP addresses. Firewalls filter forbidden packets. The network devices are linked via a model that efficiently enables directing, storing, dropping, tagging, and collecting statistics about packets status in the network. Certain network devices, such as routers [21, 16], do critical jobs as they control the network. This allows routers to calculate and specify directions of packets in the network. Surely distinct networks have distinct properties and functions.

Software-Defined Networks (SDNs) [10] are networks built via the controller-switch technique. A specific execution of this style is OpenFlow [2] useful to do different network-wide functions like balancing switch load, monitoring data flow, network management, managing devices usage, reporting service problems, host movement, and moving data stores. Hence SDNs led to existence of network programming languages [18, 19, 17, 11].

The Open Networking Foundation [33], presented in 2011 the idea of withdrawing the control managed by distinct network devices and rather adding, a new general-purpose device, controller, to manage distinct network devices and questioning packets moving in the network. The effect of this simple change is very wide; huge networks do not include complex, special-purpose, and expensive switches today. In these networks, simple programmable switches are used and programmed to optimize and configure networks. This is done by producing programs [20] executed on controllers.

This paper introduces ObjNet, an object-oriented network high-level programming language. ObjNet enables classes for controllers to manage other network devices like switches. ObjNet has an organized and simply-structured structure built on classical concepts of object-oriented coding that enables establishing rich and strong network function in a modern way. ObjNet is considered as a generalization of Frenetic [24], a network programming language of the functional type. This is obvious by the idea that the main body of codes in ObjNet and Frenetic is a query output in the shape of a sequence of data such as switches IDs, packets, etc. Statements for managing packets in ObjNet can install and establish (inserting to management laws of switches) switch laws. ObjNet enables writing simple codes to achieve complex dynamic jobs such as verification and load balancing. ObjNet codes may also resolve data and historical patterns of traffic.

Motivation

The motivation of this paper is to provide a simple structure for an object-oriented network programming language. Also another motivation is to provide a precise semantics for the proposed language model. Such semantics is rarely presented for related existing languages.

Contributions

Contributions of the current paper are the following.
1. A novel simply-built syntax for an object-oriented network programming language; ObjNet.
2. A precise semantics (in the form of states and explanations) for ObjNet programs.

Organization

The rest of this paper is organized as following. Section 2 reviews related work. Section 3 introduces the structure of ObjNet. A simple semantics to ObjNet constructs is given in Section 4.

Manuscript received October 5, 2015
Manuscript revised October 20, 2015

2. Related Work

Work most related to that presented in the current paper is reviewed in this section [3,5,8,15,20,28]. NOX [14] which is based on [13] and 4D [12] is among early attempts to design software-defined networking (SDN).

Frenetic [32, 33] is a common programming language for networks which has two main parts. The first part is a group of constructors that are kernel-level. The constructors' objective is to develop and direct streams of network packets. These constructors also are built on ideas of declarative database query-languages and functional programming (FP).

Moreover the constructors are basic for a race-free semantics, a modular design, a cost control, a declarative design, and a single tier programming. The other part of Frenetic is a dynamic platform. This platform enables all of the functionalities of removing and adding rules of the low-level type to and from action logs of routers. the language introduced in this paper, ObjNet, has an advantage over Frenetic which is ObjNet is object oriented. Therefore ObjNet can be realized as an introduction to the use of new concepts of network programming languages such as objects and classes and context-oriented network programming languages.

On the router-level, the main idea of NOX is to benefit from callbacks and explicit ideas for data-processing. Load balancer [11] and the work in [9, 10] are examples of applications that use NOX. There are many ways to improve techniques of programming networks such as Maestro [7] and Onix [8], using parallelization and distribution to support high scalability and performance

Energy saving data-movement with power control actions is built on path detection system for mobile ad-hoc networks. The router component of networks is programmable using various platforms such as Open-Flow platform. Examples of other platforms are RouteBricks [37], Shangri-La , Snortran [35] and Bro [36], and FPL-3E [38], Click modular router [34]. The concept in Shangri-La [33] and FPL-3E [38] is to build specific hardware for data-processing using high-level software that does data-processing. In RouteBricks [37], low-level computers are used to boost performance of program actions. As a functional technique, the technique of Click functional router [34], enables programming network systems.

Other parts of program network systems though high-level components are NDLog and NetCore [6]. NetCore is built on an elusive view of the complete network. NDLog is built in a completely distributed fashion. NDLog [30, 31], which extended Datalog, was introduced to fix and program code techniques of routing [29], huge networks, and approaches like index tables of parallel systems. ObjNet (presented in this paper), NDLog , and Frenetic are seen as high-level network programming languages. Although NDLog main concern is completely networks and routing protocols, Frenetic in a functional style and ObjNet in an object-oriented style treat and implement data processing such as changing header parts. Hence ObjNet equips a network administrator with a complete view of the network system unlike what is possible in NDLog and Frenetic. This is so because a program in NDLog is a unique query that is executed on every switch of the network. One advantage of network programming languages (ObjNet) is saving switch energy .

Static semantics of ObjNet have many potential network applications. The K-random algorithm in ad hoc networks using static semantic packet duplication [1] is verifiable using static semantics. The verification of static information retrieval in P2P networks is done via static semantics platform [2]. Static configurations of networked systems can be done via static semantic having the style of interoperability framework [3] such as those presented in this paper. Spreading activation with latching statics can be done in attractor networks via automatic static semantic similar to that presented this paper.

The system in [30] treats software routers in the style of Linux kernel code. In order to detect intrusions and preserve network security, Bro [36] and Snortran [35] provides tools for coding modifying strategies and robust data-filtering. ObjNet, the language presented in this paper, has the advantage over all the related languages that ObjNet does not focus on controlling a single device. This overcomes a common disadvantage of most similar network languages.

3. ObjNet: Object-Oriented Programming Language for Networks

For SDN networks, this section introduces the structure and a static-precise semantics of ObjNet, a high-level object-oriented programming language. The language uses the switch-controller technique. Figure 1 presents the constructs of ObjNet.

A program in ObjNet is a sequence of class definitions and classes of queries followed by set of commands. The output of every query is an object of an event class containing a set of values. The event class is not utilized in any previous programming language of networks including Frenetic. This is so as an event in Frenetic is composed of an infinite set of packets. An event output (values of of class objects) could be a packet, an integer, a triple of a Boolean value, a switch ID, and a switch ID, or a pair of two values. Each event is supposed to belong to a class object. In this paper, the emphasize is on the details of implementing statements using the object-oriented programming concepts.

$$
\begin{aligned}
&x, y \in lVar, \quad n \in \mathbb{Z}, \quad Q \in Classes\ of\ Quieries\ \ C \in Classes, and\ \ O \in class\ Objects.\\
&en \in events ::= n \mid ObjLift\ (Obj\ x, fun(t)) \mid ObjApplyL\ (Obj\ x, fun(t))\\
&\qquad \mid ObjApplyR\ (Obj\ x, fun(t)) \mid ObjMerge\ (Obj\ x, Objy)\\
&\qquad \mid ObjMergeF\ (L, Obj\ x, Objy) \mid ObjMergeS\ (L, Obj\ x, Objy)\\
&\qquad \mid ObjOnce\ (Obj\ x) \mid ObjMakeForR\ (Obj\ x) \mid ObjMakeR\ (Obj\ x).\\
&e \in Expr ::= l \mid en_1\ i_{op}\ en_2 \mid \&en \mid new\ en \mid ObjModify(en) \mid compute\ en \mid Objcast\ \ en\\
&\qquad \mid ObjCast < switch(s_j, C_j) \rightarrow switch(s_i, c_i) > en \mid O.ObjSend\ (Obj\ x)\\
&\qquad \mid O.ObjRegister\ (Obj\ x).\\
&S \in Stmts ::= skip \mid e \mid class\ C\ O \mid O.x := e \mid O.Addrules \mid S_1; S_2\\
&\qquad \mid if\ Q\ \emptyset\ \ then\ O.e_1\ else\ e_2 \mid while\ Q\ do\ e.\\
&C \in Eventclass ::= class\ e\{x^*, e^*, S^*\}\\
&E \in Def ::= \epsilon \mid x := Q \mid EE\\
&p \in Progs ::= (C^*, E, S)
\end{aligned}
$$

Fig. 1. The programming language model, ObjNet

4. ObjNet Semantics

The presented semantics is operational. The query component of a network language is useful for providing an up-to-date status of the network. ObjNet queries are composed classes of statements for treating packets by

1. collecting using header fields values,
2. diving data using arrival time or header fields values,
3. filtering data content of the network using a specific pattern,
4. abbreviating outputs of other queries, and
5. concluding outputs in terms of number or size of packets.

There are many operations that may be applied by a specific device on a specific packet such as ObjSendall, ObjSendcontroller, ObjSendout, or ObjChange(Obj o). The operation ObjSendcontroller sends a message to the controller to manage it. The operation ObjSendall forwards the data to all other devices. The operation ObjSendout forwards the data the switch via a specific port. The operation ObjChange(Obj o,n) changes the header field of packets that were under the actions of the object o to a new value n. A rule according to our proposed semantics is a ObjShape associated with an ObjOperation where the ObjShape is a specification that precisely illustrates a group of packets and an ObjOperation is the operation to be applied on elements of packets group. Actions are hosted by tables (called action tables) of switches. ObjIntial-rule stands for an initial setting for rules of flow tables controlling switches.

A configuration in the semantics of ObjNet has many components. The first component is a list of the class objects defined and used at this execution state of the program. The second component expresses the current contents of program variables and therefore could be a function from the set of program variables to the collection of class objects, events, and rule lists. It is worth noting that ObjNet variables may host class objects of events or rule lists. A third component of a configuration is the current values of switches flow tables. This last component can be expressed as a function from device IDs to rule tables. A fourth component of the configuration is an initial setting for flow tables of switches which have not been certified yet.

ObjNet has several types of statements. The statement class C O defines an object O of a class of C. The statement O. Addrules adds the switch rules hosted in the object O to the store of rules initially specified. The added rules are associated with switches but not adapted for application yet. The statement O. ObjRegister (Obj x) treats the action of register rules of the object O into the object x. This will result in truing the rules x permanent upon adding them to the tables of switches. The command O. ObjSend (Obj x) sends certain messages for certain actions provided by the events if the object . Similar explanations illustrate other constructs of ObjNet.

Acknowledgements

The author acknowledges support by the Al Imam Mohammad Ibn Saud Islamic University (IMSIU) (Grant number 350918).

References

[1] X. Cao, M. Klusch. Dynamic Semantic Data Replication for K-Random Search in Peer-to-Peer Networks. NCA, 2012, p. 20–27.

[2] A. Kamoun, S. Tazi, K.l Drira. FADYRCOS, a semantic interoperability framework for collaborative model-based dynamic reconfiguration of networked services. Computers in Industry (CII), 63(8),2012, pp. 756–765.

[3] Xianneng Li, Kotaro Hirasawa: Continuous probabilistic model building genetic network programming using reinforcement learning. Appl. Soft Comput. (ASC) 27:457-467 (2015).

[4] A. Eftychiou, B. Vrusias, N. Antonopoulos. A dynamically semantic platform for efficient information retrieval in P2P networks. IJGUC 3(4), 2012, pp. 271–283.

[5] Farzad Tashtarian, Mohammad Hossein Yaghmaee Moghaddam, Khosrow Sohraby, Sohrab Effati: ODT: Optimal deadline-based trajectory for mobile sinks in WSN: A decision tree and dynamic programming approach. Computer Networks (CN) 77:128-143 (2015).

[6] I. Lerner, S. Bentin, O. Shriki. Spreading Activation in an Attractor Network With Latching Dynamics: Automatic Semantic Priming Revisited. Cognitive Science (COGSCI), 36(8), 2012, pp. 1339–1382.

[7] B. Loo, J. Hellerstein, I. Stoica, and R. Ramakrishnan. Declarative routing: Extensible routing with declarative queries. SIGCOMM, 2005, pp.289-300.

[8] Monirehalsadat Mahmoudi, Xuesong Zhou: Finding Optimal Solutions for Vehicle Routing Problem with Pickup and Delivery Services with Time Windows: A Dynamic Programming Approach Based on State-space-time Network Representations. CoRR abs/1507.02731 (2015).

[9] T. Serbanuta, G. Rosu, and J. Meseguer. A rewriting logic approach to operational semantics. Inf. Comput., 2009, pp.305–340.

[10] Z. Cai, A. Cox, and T. Ng. Maestro. A system for scalable OpenFlow control. Technical Report TR10-08, Rice University, Dec 2010.

[11] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. Onix: A distributed control platform for large-scale production networks. OSDI, Oct 2010.

[12] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. ElasticTree: Saving energy in data center networks. NSDI, Apr 2010.

[13] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown, and R. Johari. Plug-n-Serve. Loadbalancing web traffic using OpenFlow. Demo at ACM SIGCOMM, Aug 2009.

[14] R. Wang, D. Butnariu, and J. Rexford. OpenFlow-based server load balancing gone wild. Hot-ICE, Mar 2011.

[15] E. Creaco, G. Pezzinga: Embedding linear programming in multi objective genetic algorithms for reducing the size of the search space with application to leakage minimization in water distribution networks. Environmental Modelling and Software (ENVSOFT) 69:308-318 (2015).

[16] A. Greenberg, G. Hjalmtysson, D. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A clean slate 4D approach to network control and management. SIGCOMM CCR 35, October 2005, pp.41-54.

[17] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. NOX: Towards an operating system for networks. SIGCOMM CCR 38(3), 2008.

[18] M. Casado, M. Freedman, J. Pettit, J. Luo, N. Gude, N. McKeown, and S. Shenker. Rethinking enterprise network control. Trans. on Networking. 17(4), Aug 2009.

[19] N. Foster, A. Guha, M. Reitblatt, A. Story, M. Freedman, N. Katta, C. Monsanto, J. Reich, J. Rexford, C. Schlesinger, D. Walker, R. Harrison. Languages for software-defined networks. IEEE Communications Magazine 51(2), 2013, pp. 128–134.

[20] Alaeddin Malek, Leila Jafarian-Khaled Abad, Samaneh Khodayari-Samghabadi: Semi-Infinite Programming to Solve Armed robot trajectory Problem using Recurrent Neural Network. I. J. Robotics and Automation (IJRA) 30(2) (2015).

[21] T. Bain, P. Campbell, J. Karlsson. Modeling growth and dynamics of neural networks via message passing in Erlang: neural models have a natural home in message passing functional programming languages. Erlang Workshop, 2011, pp. 94-97.

[22] A. Elsts, L. Selavo. A user-centric approach to wireless sensor network programming languages. SESENA 2012, pp. 29–30.

[23] T. Suzuki, K. Pinte, T. Cutsem,W. De Meuter, A. Yonezawa. Programming language support for routing in pervasive networks. PerCom Workshops,

[24] S. Hong, Y. Joung. Meso: an object-oriented programming language for building stronglytyped internet-based network applications. SAC 2013, pp.1579–1586.

[25] C. Monsanto, N. Foster, R. Harrison, D. Walker. A compiler and run-time system for network programming languages. POPL 2012, pp. 217–230.

[26] J. Rexford. Programming languages for programmable networks. POPL 2012, pp. 215–216.

[27] H. Arneson, C. Langbort. A linear programming approach to routing control in networks of constrained linear positive systems. Automatica 48(5), 2012, pp. 800-807.

[28] Seyed Morteza Hatefi, Fariborz Jolai, S. Ali Torabi, Reza Tavakkoli-Moghaddam: A credibility-constrained programming for reliable forward-reverse logistics network design under uncertainty and facility disruptions. Int. J. Computer Integrated Manufacturing (IJCIM) 28(6):664-678 (2015).

[29] B. Loo, J. Hellerstein, I. Stoica, and R. Ramakrishnan. Declarative routing: Extensible routing with declarative queries. SIGCOMM, 2005, pp. 289-300.

[30] B. Loo, T. Condie, J. Hellerstein, P. Maniatis, T. Roscoe, and I. Stoica. Implementing declarative overlays. SIGOPS 39(5), 2005, pp 75-90.

[31] N. Foster, R. Harrison, M. Meola, M. Freedman, J. Rexford, and D. Walker. Frenetic: A high-level langauge for OpenFlow networks. PRESTO, Nov 2010.

[32] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek. The Click modular router. ACM Transactions on Computer Systems 18(3), Aug 2000, pp 263-297.

[33] N. Foster,R. Harrison, M. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker. Frenetic: A Network Programming Language.the 16th ACM SIGPLAN international conference on Functional programming,2011 pp. 279–291.

[34] S. Egorov and G. Savchuk. SNORTRAN: An Optimizing Compiler for Snort Rules. Fidelis Security Systems, 2002.
[35] M. Dobrescu, N. Egi, K. Argyraki, B. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy. RouteBricks: Exploiting parallelism to scale software routers. SOSP, Oct 2009.
[36] V. Paxson. Bro: A system for detecting network intruders in realtime. Computer Networks 31(2324), Dec 1999, pp. 2435-2463.
[37] P. Gao, W. Shi, H. Li, W. Zhou. Indoor Mobile Target Localization Based on Path-planning and Prediction in Wireless Sensor Networks. WSEAS Transactions on Computers 12(3), 2013, pp. 116–127.
[38] M. Chen, X. Li, R. Lian, J. Lin, L. Liu, T. Liu, and R. Ju. Shangri-la: Achieving high performance from compiled network applications while enabling ease of programming. PLDI, Jun 2005, pp 224-236.
[39] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. SIGCOMM CCR 38(2), 2008, pp. 69-74.
[40] V. Bhanumathi, R. Dhanasekaran. Energy Efficient Routing with Transmission Power Control based Biobjective Path Selection Model for Mobile Ad-hoc Network. WSEAS Transactions on Computers 11(11), 2012, pp. 407–417.
[41] The Open Networking Foundation, Mar 2011. See http:// www.opennetworkingfoundation.org/
[42] M. Cristea, C. Zissulescu, E. Deprettere, and H. Bos. FPL-3E: Towards language support for reconfigurable packet processing. SAMOS, Jul 2005, pp 201-212.
[43] J. Harding. Decidability of the Equational Theory of the Continuous Geometry CG(F). J. Philosophical Logic 42(3), 2013, pp. 461–465.
[44] E. Golemanova. Declarative Implementations of Search Strategies for Solving CSPs in Control Network Programmings. WSEAS Transactions on Computers 12(4), 2013, pp. 174–183.