Using DAIM as a Reactive Interpreter for OpenFlow Networks to Enable Autonomic Functionality

Pakawat Pupatwibul, Ameen Banjar, Robin Braun
CRIN, University of Technology Sydney, Australia.
{ndomon66, dr.banjar}@gmail.com, robin.braun@uts.edu.au

ABSTRACT
OpenFlow is the first standardization of Software Defined Networks. OpenFlow approach, however, has number of limitations: it restricts its use within a single-domain, it is not scalable, and it does not adapt well to changes in local environments. We evaluate the number of approaches to solve these limitations, and propose DAIM model (Distributed Active information Model) which can be integrated into the OpenFlow structure at the level of the switches to provide a reactive interpreter that will manage the flow tables autonomically.

Categories and Subject Descriptors
C.2.1 [Computer-Communication Networks]: Network Architecture and Design – Network communications, Network topology

Keywords
OpenFlow, Distributed systems, Autonomic functionality

1. INTRODUCTION
In the last few years network technologies have been increasing significantly in performance, complexity, functionality, driven by the needs of the modern world. This has given rise to a new network paradigm called Software-Defined Networking (SDN). SDN separates the forwarding plane from the control plane, and in so doing enabling the creation of a standardized programming interface [1]. Flow computation is done in a centralized controller with the switches only performing simple forwarding functions. OpenFlow is very efficient at moving the computational load away from the forwarding plane and into a centralized controller. This centralization brings optimality, but creates additional problems of its own including single-domain restriction, scalability, robustness, and the ability for switches to act autonomously.

In this poster, we are proposing to situate DAIM (Distributed Active Information Model) clouds within the switches, and have these clouds compute the flows from local information and centralized requirements databases. The proposal moves the computational load to the switches and effectively the DAIM clouds act as reactive interpreters [2]. The functionality of the Controller migrates into a Requirements Database while its computational obligations migrate into the DAIM clouds on the switches. The behaviour of the network is then simply changed through changes in the requirements database. Moreover, the decision making ability will be local within each switch, on the basis of collected information by intelligent agents, which allows it to autonomically adapt according to the ever-changing circumstances.

SIGCOMM'13, August 12–16, 2013, Hong Kong, China.
ACM 978-1-4503-2056-6/13/08.

2. RELATED WORKS
There are three approaches to resolve the limitations of OpenFlow such as optimizing the centralized controller, empowering the switches to process some control functions, and distributing the control platform [3]. However, first approach still relies on only a single controller and sends batching messages to the switches not in a runtime configuration. This can significantly affect the packet forwarding process within large-scaled networks. Furthermore, OpenFlow is more flexible since its control logic can realise behaviours that cannot be easily achieved by a set of policy rules installed in authorized switches [4], [5]. Finally, synchronization within distributed controllers should be in a runtime mode in order to provide reliability and robustness [6].

3. DAIM MODEL ARCHITECTURE
The DAIM model is implemented within each OpenFlow switch using a Multi-agent operating system, which is supported by DAIM agents as a field of distributed active artificial intelligent (AI) to enable autonomic functionality. DAIM model composed of a system requirement database (SRD) and DAIM agents. These components use augmented OpenFlow protocol called DAIM protocol for corresponding messages between them. Intelligent DAIM agents which reside in each switch as an independent computational environment are implemented in a Java Virtual Machine (JVM). In addition, they interact with the database and neighbouring switches to exchange information. Therefore, DAIM agents can compute their own local decisions according to the business needs defined in the database. The DAIM system requirement database schema holds business needs and the network information such as host identifier, business requirements, topology discovery, QoS, connectivity, bandwidth, users, and global view of the entire network (see Figure 1). DAIM model uses this information to make management and routing decisions. The monitoring data includes changing of network links, network topology, changing of host locations, so that would facilitate the calculation and installation of shortest route.

The basic information unit of each DAIM agent includes *Attributes, Method behaviors, Algorithms,* and *Messaging.* These four characteristics can modify the value of its own property. This inverts the traditional *get* → *compute* → *set* process in network management. Instead, property values are change by either calling methods, or by the modules proactively calling processes that compute the required values, and then call those methods.

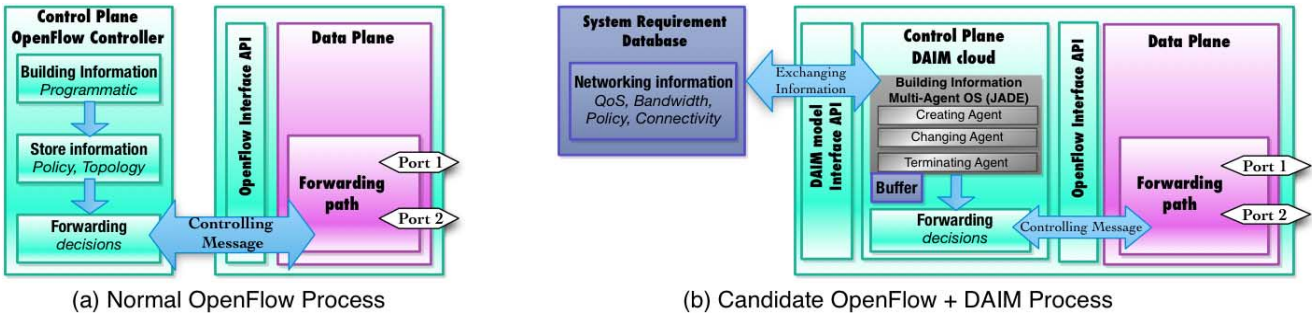

Figure 1. DAIM model architecture as an intelligent computational environment

4. PACKET PROCESSING WITHIN DAIM

The DAIM cloud has a multi-agent operating system such as JADE (Java Agent DEvelopment Framework) that can create, change, and terminate the intelligent DAIM agents. These agents have the responsibility to maintain their own values, and they can adapt and modify their own value according to the collected information. DAIM agents can make their own local decisions based on the system requirements.

When the DAIM cloud receives an unmatched packet, it creates DAIM agents which can access and control network elements such as system requirement database, and other switches to determine the forwarding rules. The DAIM agents should be able to check this flow against system requirements and other policies to see whether it should be allowed, and if allowed the DAIM agent needs to compute a path for this flow, and install flow entries on every switch along the chosen path. Finally, the packet itself will be forwarded. The DAIM agents can provide a distributed environment where the network information is the property (values) of software agents residing in virtual machines that are distributed throughout the network elements. Therefore, the DAIM agents have the ingredients to implement autonomic behaviours.

Table 1. Comparison of normal and candidate processes

Normal OpenFlow process	Candidate OpenFlow+DAIM process
Unknown packet arrives at a switch	Unknown packet arrives at a switch
Switch cannot match a flow table entry	Switch cannot match a flow table entry
Switch uses OpenFlow protocol to forward pack to controller	Switch instantiates a new agent, which "owns" a new row in the flow table.
Controller computes forwarding destination for the packet.	This is a unique Agent (which "owns" that row in the flow table). It computes the forwarding destination for the packet according to the system requirement database and exchanging information between agents
Controller uses OpenFlow protocol to update flow table entry on the switch, which now knows how to forward the packet, and similar ones	The agent updates the flow table row using its built in "methods"

Table 1 shows a comparison between normal OpenFlow processing and the candidate DAIM model within OpenFlow which typically depend on the system requirement database and the intelligent DAIM agents. Therefore, individual switch can serve any coming packets locally.

5. CONCLUSION

In this poster, we propose the DAIM model within OpenFlow switches as the compiled reactive interpreter paradigm. DAIM is a new way of viewing information objects that manage the behaviour of a network, or any other complex distributed electronic environment. The DAIM cloud is proposed with the hope to address the limitations of current approaches and future distributed network systems aiming at an autonomic computing management strategy.

6. ACKNOWLEDGMENTS

This poster is sponsored by the Centre for Real-Time Information Networks (CRIN) in the Faculty of Engineering & Information Technology at the University of Technology, Sydney (UTS).

7. REFERENCES

[1] ONF White Paper. Software-Defined Networking: The New Norm for Networks. April 2012.

[2] R. Braun and F. Chiang. A Distributed Active Information Model Enabling Distributed Autonomics in Complex Electronic Environments. In *Broadband Communications, Information Technology & Biomedical Applications*, November 2008.

[3] Z. Cai, A. L. Cox, and T. E. Ng. Maestro: A System for Scalable OpenFlow Control. In *Rice University Technical Report*, 2010.

[4] A. R. Curtis, et al. Devoflow: Scaling flow management for high-performance networks. *ACM SIGCOMM-Computer Communication Review*, vol. 41, 2011.

[5] M. Yu, et al. Scalable flow-based networking with DIFANE. *ACM SIGCOMM Computer Communication Review*, vol. 41, pp. 351-362, 2011.

[6] A. Tootoonchian and Y. Ganjali. HyperFlow: A distributed control plane for OpenFlow. In *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, 2010.