Declarative, Distributed Configuration[1]

Sanjai Narain, Dana Chee, Chung-Min Chen, Brian Coan, Ben Falchuk, Dov Gordon, Jon Kirsch, Siun-Chuon Mau, Aditya Naidu, Simon Tsang, Applied Communication Sciences, USA

Sharad Malik, Shuyuan Zhang, Princeton University, USA

It is well documented that configuration errors account for 50% to 80% of downtime and vulnerabilities in networks. The Assured and Dynamic Configuration(ADC) system [1, 2, 3] offers a set of fundamental tools to help eliminate such errors. These tools are for requirement specification, configuration synthesis, diagnosis, repair, verification, moving-target defense and symbolic reachability analysis. These exploit the power of modern SAT and SMT solvers and are being transitioned to real enterprises. ADC's specification language contains fundamental logical structures and relationships associated with different protocols. It formalizes the idea that network security and functionality requirements as specified, for example, in architecture diagrams, can be regarded as a superposition of such logical structures and relationships. These are modeled as constraints on configuration variables within and across components. Superposition is accomplished with Boolean operators.

A SAT or SMT solver is used to solve requirements to compute values of configuration variables satisfying all requirements. In contrast, traditional configuration languages force one to specify the value of every configuration variable, in effect, forcing one to manually solve the constraint satisfaction problem.

As shown in Figure 1. ADC System ArchitectureADC's overall architecture is the same as that of Software-Defined Networking (SDN). It assumes a global controller at which one can design and generate the configurationfor the network as a whole. It also assumes an out-of-band network over which the controller monitors the current configurations and downloads new ones to components.

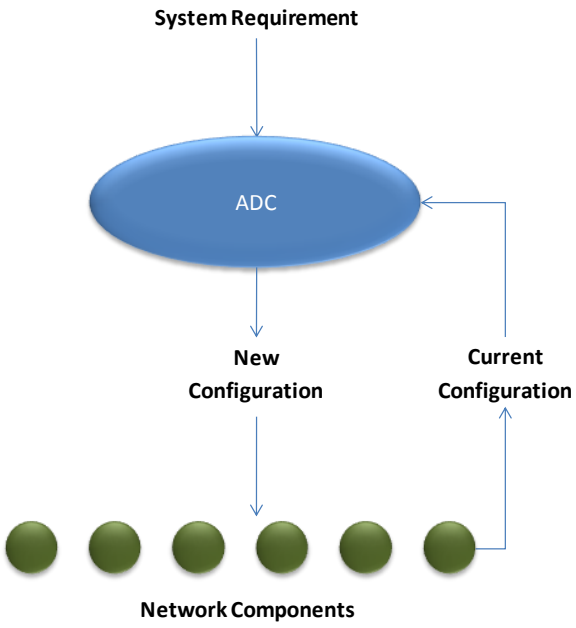

Figure 1. ADC System Architecture

The major difference with SDN is that ADC assumes full-featured routers, not just those that do forwarding and access-control. Thus, ADC does not require one to reimplement the well engineered routing, security and performance management protocols available in modern routers. In fact, it lets one take full advantage of these by letting one just correctly configure these and have them do the "heavy lifting". For example, cryptographic separation of multiple fault-tolerant virtual private networks can be accomplished by configuring IPSec, GRE, RIP and OSPF [1, 2].

The Distributed ADC (DADC) system removes the assumptions of a centralized controller and out-of-band control network. Decentralization is

[1] This work has been funded by Air Force Research Laboratory under contract FA8750-13-C-0030. "Approved for Public Release; Distribution Unlimited : 88ABW-2014 1877 22 April 2014"

accomplished by a new Configuration-Agreement Protocol (CAP) based on the total ordering guarantees of group communication protocols such as JGroups, and the determinism of SAT and SMT solvers.

The challenge of in-band configuration is maintaining the invariant that the controller does not lose connectivity to a node before it has reconfigured it. This invariant is encoded as a constraint on current and final static routes and solved by a SAT or SMT solver to compute a safe reconfiguration order. If only dynamic routing is used then no constraint solver is used. The reconfiguration order is computed from a reverse breadth-first search traversal of the network as a tree with the controller as root.

As shown in Figure 2, the set of network components is partitioned into enclaves each controlled by a DADC controller. Each controller has the full functionality of a centralized ADC server. Controllers communicate with each other over a CAP bus. Also communicating over this bus are Enterprise Management Systems and Intrusion Detection and Response Systems that provide information about the dynamic state of components: up, down, compromised.

CAP guarantees that messages are delivered to all controllers in the same order. Therefore it presents to each controller an identical view of the dynamic state of all components. Each controller also has the identical System Requirement governing the whole network. Upon receipt of a message, each controller solves the System Requirement in the context of the current dynamic state. Since SAT or SMT solvers that we use are deterministic, each controller arrives at identical conclusions about the new configurations of all components, not just its own. Each controller then applies configurations relevant to its enclave to the enclave components, and the entire network converges to a new configuration satisfying System Requirement.

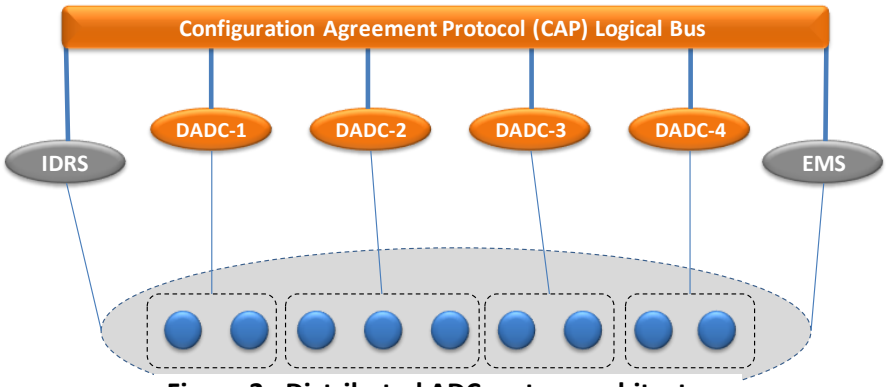

Figure 2: Distributed ADC system architecture

We are currently exploring the application of DADC ideas to SDN and vice versa. For example, a DADC-like domain-specific constraint language and SAT/SMT solvers could simplify controller programming. CAP ideas could be used to design distributed SDN controllers. DADC's in-band configuration algorithm could be used to design similar algorithms for SDN. DADC could be used to manage a hybrid legacy and SDN network. Conversely, SDN languages like Frenetic [4] may be used to improve DADC's specification language.

References

1. Sanjai Narain, Sharad Malik. ConfigAssure: A Science of Configuration. Proceedings of MILCOM, 2013, San Diego, CA.
2. Sanjai Narain, Dana Chee, Sharad Malik. Demonstrating Assured and Dynamic Configuration Over a Live, Emulated Network.pdf
3. Sanjai Narain, Sharad Malik, Ehab Al-Shaer. Towards Eliminating Configuration Errors in Cyber Infrastructure. Proceedings of IEEE Symposium on Configuration Analytics and Automation, Arlington, VA, 2011.pdf
4. Frenetic: A Family of Network Programming Languages. http://www.frenetic-lang.org/