Efficient Deployment of Network Management Policy Using Distributed Database Abstraction

Kamran Ali Akhtar (Student)*, Muhammad Shahbaz (Student)†, Saad Qaisar*
*National University of Sciences and Technology †Georgia Institute of Technology
{12msitkakhtar, saad.qaisar}@seecs.edu.pk, shahbaz@cc.gatech.edu

Abstract. A network switch flow table can be seen as a relational database, it contains records having attributes belonging to data flows, such as, switch port, header fields (*i.e.*, Ethernet and IP source/destination) and actions. This flow table has same properties like any other database table and required CRUD (Create, Read, Update and Delete) operations that are basic operations performed on any table in a database. With OpenFlow, these CRUD operations are performed by an external controller and are seen as transactions. These transactions should have ACID (Atomicity, Consistency, Isolation and Durability) properties like in a normal database. A network that has multiple switches can be considered, operating as a distributed database. If we use a distributed database engine in the controller and use SQL for querying these databases, we can leverage benefits implemented in distributed database engines with SQL's simple syntax.

Distributed database management systems (DDBMS) is a mature field and has well known industry standards and implementations available from propriety and open-source communities [2]. Distributed database, fragments data on multiple nodes and promises transparent management of distributed data, easier system expansion, reliability through distributed transactions and improved performance. DDBMS's have already defined concurrency control mechanisms with different levels of serializability, centralized and distributed two phase locking, deadlock prevention, avoidance, detection and resolution. These also have mechanisms for handling transaction, site, media, and communication failures. Distributed databases have layers of query processing: (1) decompose the query, (2) optimize the query, and (3) perform distributed query execution and data optimization.

The mechanisms of distributed data reliability, consistency, scalability and concurrent management of multiple data sources with DBMS is well established. Achieving the same with a controller like POX maybe difficult courtesy dependance on switch generated reactive messages. It has fewer capabilities to detect transaction, switch and media failures. Distributed databases are implemented with reliable protocols to overcome these failures. The distributed query processing capabilities that include query composition, localized data management, and deadlock prevention and detection schemes can be used to decrease the working load on northbound controllers like Pyretic, Frenetic [1].

For, switch to controller, events and flow statistics, POX uses OpenFlow generated events and has to deploy flow rules that gather these statistics. These flow rules should be placed in proper order so that before leaving the switch the packet should be processed by statics gathering flows. If these flows are not placed in proper order then statistics are not gathered, this problem can be solved by using sFlow that is also an industry standard protocol available in most of the legacy and SDN switches.

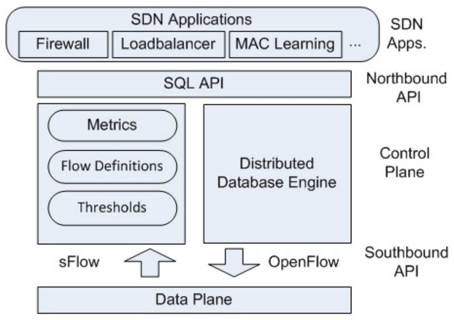

Figure 1: Architecture of SQL Controller.

For leveraging benefits like code-base and database management strategies with well known SQL API, we propose a controller platform that will use SQL API as northbound API and deploy flow rules through distributed database engine, as shown in Figure 1. With industry standard SQL API, query would be checked and validated for errors. This query will be executed by distributed database engine that will decompose the query for deployment on network switches with predefined network policy. With multiple queries, distributed database engine will check the queries for deadlocks by applying serializability and execute the queries by schemes like 2P locking. It will perform check pointing and record recovery information for solving any failures during the transaction. It will use the OpenFlow protocol to deploy a network flow on the switch. For querying network state, we propose to use sFlow that will gather the statistics and can also perform packet inspection on the wire-speed not leaving the data plane. It will store results in a database that can be queried with SQL API.

We provide a demonstration of our SQL framework for MAC learning, firewall and load-balancer applications. We use POX as a backend controller and add a layer that accepts SQL queries, parses them and deploy flow rules accordingly. We test the relative performance of our controller using mininet emulator. Our implementation decreases the line-of-code by 50% and demonstrates ease-of-use compared to traditional controllers.

1. REFERENCES

[1] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker. Composing software defined networks. *NSDI, Apr*, 2013.

[2] M. Tamer èOzsu and P. Valduriez. *Principles of distributed database systems*. Springer, 2011.