Improving Quality of Service in Cloud Based Networks with Software Defined Networks

Chinthagunta Mukundha

Professor, CSE Department, Institute of Aeronautical Engineering, Hyderabad -500043, Andhra Pradesh, India

Abstract: ***With the increase of network in cloud the complexity and flexibility of network control and management becomes a nontrivial problem. Both Software Defined Network (SDN) and Autonomic Network technologies are sophisticated technologies for the network control and management. In AQSDN, the various QoS features can be configured autonomically in an OpenFlow switch through extending the OpenFlow and OF-Config protocols. Based on AQSDN, a novel packet context-aware QoS model (PCaQoS) is also introduced for improving the cloud network QoS. PCaQoS takes packet context into account when packet is marked and managed into forwarding queues. The implementation of a video application's prototype which evaluates the self-configuration feature of the AQSDN and the enhancement ability of the PCaQoS is presented in order to validate this design.***

Keywords: Cloud network, SDN, Autonomic Management, Context aware, Quality of Service

1. Introduction

In the Cloud based networks increment of various network equipments and services, the complexity of network control and management has become more complex. The QoS management is an important part of network management. Various QoS models and mechanisms have been proposed, but there is no common algorithm. The advantages and disadvantages of the different combinations of queue management and scheduling, which works at various workloads and different transmission mediums are analyzed.

In order to reduce the Cloud network management cost and the probability of Cloud network failure, autonomic network technologies is introduced in Cloud network control and management, and the autonomic function is also considered as an important component in many projects which focused on future network architecture. Software Defined Network (SDN) separates the control plane and the data plane, some Network control and management functions could be implemented through the software instead of updating huge numbers of network elements.

OpenFlow protocol which proposed by Open Networking Foundation (ONF) is the most popular standard for communication between controller and OpenFlow switch in SDN. In order to support QoS management functions, each version of OpenFlow protocol defines the enqueue action, but the queue configuration has not been involved.

We design an Autonomic QoS management mechanism in SDN (AQSDN) for Cloud network QoS guarantee. In this mechanism, the controller undertakes the function of analysis and decision in the autonomic control loop, while the switch acts as collector of the context and executive of the behavior. This mechanism controls adaptively the QoS rules according to the context from data plane and the policy from the operator. In addition, for improving the quality of multimedia service, we also propose the Packet Context-aware QoS model (PCaQoS), which processes packets according to their semantic precedence level.

2. Theoretical basis and Literature Review

Very little research is done in the field of language of SDN because this is a relatively new field. OpenFlow is the Application Programming Interface used for the controller to talk to the switches below where as the SDN makes it possible to program the network, it does not make it easy. OpenFlow controllers in today's generation offer low-level APIs that mimic the underlying switch hardware. As such a new platform must be created to provide programmers to program with ease and not have to deal with the lower level switches.

Cloud computing refers to the delivery of computing resources over the Internet. Instead of keeping data on your own hard drive or updating applications for your needs, you use a service over the Internet, at another location, to store your information or use its applications. Doing so may give rise to certain privacy implications. The cloud computing model allows access to information and computer resources from anywhere that a network connection is available. Cloud computing provides a shared pool of resources, including data storage space, networks, computer processing power, and specialized corporate and user applications.

An automatic and scalable QoS control Architecture adds Rate-limiters and queue mapping API to Open-Flow. The rate-limiters API is used to map one or more flows to the special rate limiters, while the queue mapping API is used to map a flow to the special priority queue. QoSFlow extends the OpenFlow protocol 1.0 for improving the QoS control capability on OpenFlow/SDN networking. PindSwitch provides Output Queue Control message for configuration of the queue scheduling.

3. AQSDN Architecture

The AQSDN Architecture is illustrated in Fig.1. The QoS control module which is located in SDN controller decides or chooses QoS rules dynamically. The OpenFlow protocol and OF-config protocol is extended for acquiring the context from SDN switch or issueing the QoS rules to SDN switch. For sensing context and instantiating the QoS rules, several functions and components are implemented in OpenFlow switch.

3.1 QoS Control Module

The QoS control module, which is an application of the ONF controller, has two function models. One is QoS scheme decision model, which determines suitable queue management and scheduling schemes as well as their parameters. This model is activated infrequently, e.g., when a switch is initialized or when the operator forces it. The other one is QoS action decision model, which determines packets marking and designates the queue for each flow adaptively. This model may be activated frequently. QoS schemes and actions are collectively referred as QoS rules. The QoS control module comprises Context manager, Analysis, Rule decision, Requirement DB, Rule DB and Policy DB.

The detail functions of these models are described as follows:

- **Requirement DB:** It stores the QoS requirements, which are provided by network operator, end user or other applications running on the controller.
- **Rule DB:** It stores the historical QoS rules which are used for the special requirements or in specified network environments.
- **Policy DB:** It stores the QoS Polices supported by OpenFlow switch, which can be provided by network operator or other applications.

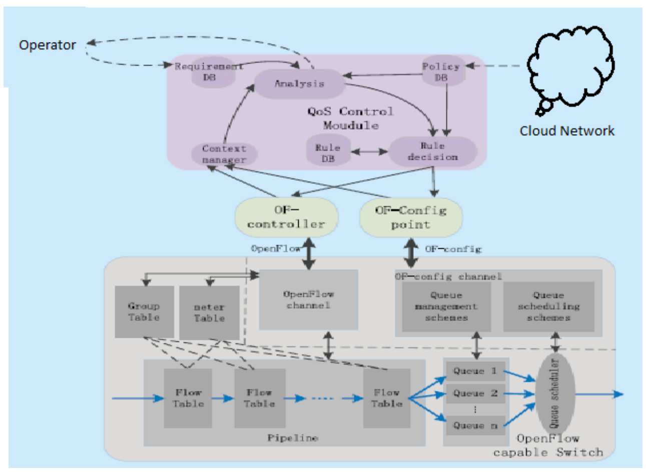

Figure 1: Architecture of AQSDN

Context manager: context manager is responsible for aggregating, storing and managing context perceived from the data plane. Generally, context could include any information charactering the situation of an entity. In AQSDN, we only focus on QoS management of the cloud network. So the entity is network and service. The network context and the Flow context are defined as Definition 1and 2.

Definition 1. (Network Context): network context is an information set characterizing the network performance, which includes the state information of the switch or node (e.g., utilization ratio of CPU and memory on a node and the length of the packet queue), and the link information, (e.g., the Packet Loss Ratio (PLR) delay and jitter, available bandwidth of the link). The controller attains the node state information through accessing the Management Information Base (MIB), and the link information through analyzing the statistical information of the flow. As we known, the statistical information are divided into multiple granularities: table leve l, flow level, port level, queue level and meter level.

Definition 2. (Flow Context): Flow context is an information set characterizing a service flow, which includes the inherent feature (e.g., the service type and the QoS requirement), and the real-time flow feature (e.g., the burst rate).The inherent features of the service can be carried in the first packet of the flow, and then be achieved by controller with Deep Packet Inspection (DPI) technologies. The controller can achieve the real-time feature through querying the switch for statistical information.

Analysis: The tasks of the analysis module are to analyze whether the QoS requests can be satisfied and if there are conflicts among them. If conflicts are founded or the QoS requests could not be satisfied, these results are sent to the administrator. If the QoS requests could be satisfied, the related contexts and requirements are forwarded to the QoS rule decision module.

Rule decision: T he Rule decision component can chooses the appropriate QoS rule from the QoS Rule DB if it exits. Otherwise, this component plans new QoS rule based on the context, QoS requirement and QoS policy stored in the QoS Policy DB, and stores the new QoS rule into the QoS Rule DB for use later. To describe the control procedure clearly, the main process of the QoS control module is presented in algorithm 1.

3.2 The enforcement of the QoS rules

For QoS action in each flow, the packet marking algorithms selection is configured via OpenFlow protocol by the OpenFlow controller. Meanwhile, the QoS schemes for each queue, queue management and schedule schemes are configured through OF-config protocol by the OF-config point.

3.2.1 The enforcement of the QoS actions

The existing OpenFlow protocol defines the message about the configuration of the flow table, the group table and meter table. The enqueue/ set-queue action specifies which queue attached to a port should be entered by a flow. And the meter table and meter band provide the meter operation for flow. However, the existing remarking bands, which only lower the drop precedence level of the packet, does not satisfy various QoS requirement. So it is necessary to extend OpenFlow protocol so that it could support diverse packet marking algorithms. Based on the definition of meter band, the band type, rate, and counters fields are fixed, while other arguments are variable with the band type. For example, there is prec level field in drop band and there is not in remarking band. On the other hand, the number of specific arguments is unpredictable for new band.

Algorithm 1 QoS_Control

```
Procedure QoS_Control ( ) {
1: while (TURE) {
2: QoS_req:=read_QR (Requirement_DB );
3: QoS_policy:=read_QP(QoS_policy_DB);
4: Context:=wait_for_context();
5: Result:=analysis(QoS_req,QoS_policy, Context)
6: if(Result==Conflict)
7: exception(THERE_IS_CONFLICT);
8: else if(Result==Not_Guarantee)
9: exception (QoS_IS_NOT_SATISFIED);
 else
10: QoS_rule:=QoS_rule_decision (QoS_req,QoS_policy,
Context);
11: store_QoS_rule(QoS_rule);
}
}
```

4. Secure and Dependable Control Platform

Here, we present the general design of the secure and dependable SDN control platform we propose.

4.1 Replication

One of the most important techniques to improve the dependability of the system is replication. As can be seen in below figure our controller is replicated, with three instances in the example. Applications should be replicated as well. Besides replicated instances of the controller, in the Fig:2 we can observe application B also replicated in all controller instances. This mixed approach ensures tolerance of both hardware and software faults, accidental or malicious. Replication makes it possible to mask failures and to isolate malicious or faulty applications and/or controllers. Moreover, in case of a network partition, application B, with the proper consistency algorithms, will still be able to program all network switches, contrary to application A.

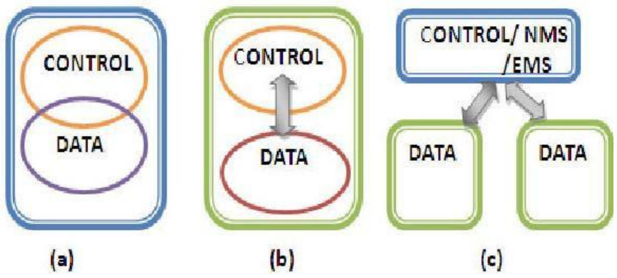

Figure 2: Secure & Dependable SDN

4.2 Diversity

Another relevant technique to improve the robustness of secure and dependable systems is diversity .Replication with diverse controllers is a good starting case. The basic principle behind this mechanism is to avoid common-mode faults (e.g., software bugs or vulnerabilities).For example, it is known that off-the-shelf operating systems, from different families, have few intersecting vulnerabilities which means that OS diversity constrains the overall effect of attacks on common vulnerabilities. In SDNs the same management application could run on different controllers. This can be simplified by defining a common abstraction for applications (a northbound API).

4.3 Self-healing mechanisms

Under persistent adversary circumstances, proactive and reactive recovery can bring the system back to a healthy state, replacing compromised components, and keep it working virtually forever. When re-placing components, it is important that the replacement be done with new and diverse versions of the components, whenever possible. In other words, we should explore diversity in the recovery process, strengthening the defense against attacks targeting specific vulnerabilities in a system.

4.4 Dynamic device association

If a switch is associated with a single controller, its control plane does not tolerate faults. Once the controller fails, the control operation of the switch fails and the switch will need to associate with another controller. For this reason, a switch should be able to dynamically associate with several controllers in a secure way A switch associated with different controllers would be able to automatically tolerate faults. Other advantages include increasing control plane throughput several controllers could be used for load balancing and reducing control delay by choosing the quickest-responding controller.

5. What Is OpenFlow?

OpenFlow is an open, standards-based communications protocol and an example of device-based SDN. OpenFlow provides access to the forwarding plane of a network switch or router over the network, facilitating more sophisticated traffic management, especially for virtualized and cloud environments. The OpenFlow protocol is standardized and managed by the Open Networking Foundation (ONF), whose mission also includes the promotion of SDN technologies as a whole.

In a classical router or switch, the data plane and the control plane reside on the device. OpenFlow enables part of control plane operations to run on external servers called controllers. In practice, an OpenFlow API is generally a feature added to commercial network devices, whose hardware architecture and features remain crucial to network performance. The standard control plane of the device remains in place and performs traditional routing or switching. Today, most OpenFlow- enabled devices can also support both OpenFlow traffic and non-OpenFlow traffic, with mechanisms for determining to which pipeline each traffic flow should be routed. The real benefit of OpenFlow lies in the applications that it can enable. New forms of traditional control plane applications such as security or specialized QoS functions—and even entirely new applications—may be written to these controllers, as shown in Fig:3 below. This will enable cloud and hosting providers, in particular, to develop and market more truly differentiated services to their clients. Traditional enterprises can also benefit from this type of third-party network application development, for example, in developing capabilities that help meet the operational or regulatory requirements of their industry.

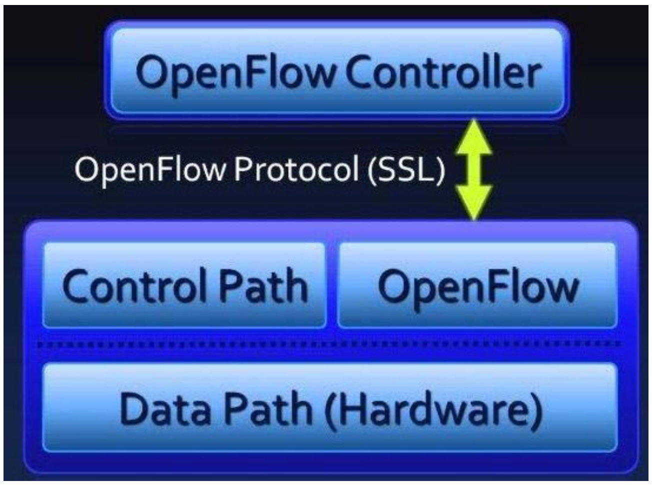

Figure 3: Controller Path

5.1 Use Cases for Control Plane Abstraction:

SDN will enable a wide variety of use cases as the technologies mature. In the near term, these are some of the most commonly envisioned scenarios:

Service assurance through flow optimization in the Wide Area Network (WAN). Public cloud providers may wish to ensure their SLAs by maintaining visibility and control of traffic all the way to the client's network edge. This can be achieved by deploying OpenFlow-enabled devices both at the cloud provider edge and client ingress, with both devices communicating to the cloud provider OpenFlow controllers. OpenFlow can also help provide granular control of inter-data center traffic, including backup or disaster recovery operations.

Service differentiation through rapid customization. As illustrated in Figure 1, the ability to develop new features quickly for highly specialized use cases is appealing to many, particularly in the cloud and hosting space, as it can provide opportunities for timely service differentiation and incremental monetization of the network. Such use cases might take the form of new security offerings, service levels, or bandwidth on demand.

Service velocity through highly scalable and easily orchestrated network virtualization. By defining within the controller a set of policies that can be applied to any number of flows at need, the operator is able to truly divorce the service delivered to the client from the physical locations of the infrastructure supporting it.

5.2 Cloud SDN

Dynamic nature of cloud services requires server virtualization to be administered in real time utilizing network virtualization. Software Defined Networking is the new paradigm of networking, which uses a centralized controller to control the flow of packets in the data plane. This new approach makes network management easier and has ability to save costs for the organization. There has been significant advancement in cloud computing technologies, which has led to the development of cloud management tools like OpenStack (An Open source Infrastructure as a Service (IaaS) cloud computing project).

6. Future Work

The future of cloud networking will rely more and more on software, which will accelerate the pace of innovation for cloud networks as it has in the computing and storage domains. SDN promises to transform today's static networks into flexible, programmable platforms with the intelligence to allocate resources dynamically, the scale to support enormous data centers and the virtualization needed to support dynamic, highly automated, and secure cloud environments. With its many advantages and astonishing industry momentum, SDN is on the way to becoming the new norm for networks.

7. Conclusion

In the traditional IP network, resource utilization improvement and network QoS guaranteeing are very complicated for cloud network operators. For solving this problem, research communities propose lot of ideas on it. One of them is to upgrade the network nodes with autonomic abilities. In addition, Software Defined Network provides the capability to implement network control and management functions by software. In this paper, the AQSDN architecture, which combines the advantages of the autonomic network management and the SDN technologies, is proposed. A novel QoS model which is called Packet Context-aware QoS (PCaQoS) model is also presented based on the AQSDN architecture.

References

[1] Sivaraman, Anirudh, et al. No silver bullet: extending SDN to the data plane. Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks. ACM, 2013.
[2] Kephart J O, Chess D M. The vision of autonomiccomputing[J]. Computer, 2003, 36(1): 41-50.
[3] Miller. B. The autonomic computing edge: Can you CHOP up autonomic computing? Technical report, IBM, 2008. http://www.ibm.com/developerworks/autonomic/library/ac-edge4.
[4] EFIPSANS project. http://www.efipsans.org/.
[5] ANA (Autonomic Network Architecture) Project. http://www.ana-project.org/.
[6] HAGGLE Project. http://www.haggleproject.org/
[7] CASCADAS project. http://www.cascadas-project.org/.
[8] BIONETS project. http://www.bionets.eu/.
[9] OpenFlow Specification 1.3.0. https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf.

Author Profile

Mukundha Chinthagunta, Associate Professor, Department of Information Technology, Institute of Aeronautical Engineering, HYD-500043, AP, India.