Software-Defined Networking in The Internet of Things

John H. Schonebaum
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
schon094@morris.umn.edu

ABSTRACT

This paper provides an overview of the Software Defined Network (SDN) architecture. The demands of modern networking, like cellular and the Internet of Things (IoT), have created a strain on traditional networking methods. Software-Defined Networking attempts to to create a new networking structure that is more centralized, flexible, and dynamic. This is accomplished by separating the control plane (the decision-making logic) of a network from the data plane (the physical routers and switches). We will give an overview of the history and purpose of SDN, then examine a proposal for an SDN controller that focuses on operating in the IoT landscape.

Keywords

internet of things (IoT), software defined networking (SDN)

1. INTRODUCTION

Increasingly, electronic devices are being designed with internet capability. The range of devices able to access the internet has expanded from laptops and desktop computers to include a wide variety of machines with varying levels of complexity and ability. For consumers, a smart home might be equipped with sensors that control room temperature over a network. Smart devices are being developed for home use that range from voice-controlled personal assistants to kitchen appliances. Sensors used in automobiles are able to transmit wireless data, leading to optimal travel patterns. This networking paradigm is collectively known as the Internet of Things.

The variety of IoT devices has created challenges that current networks can not adequately address. These devices have a broad range of hardware and software limitations, communication interfaces, and sensor capability. The number of IoT devices is growing and shows no sign of stopping. In 2017, the number of IoT devices increased by 31% to 8.4 billion. By 2020, it is projected to grow to 30 billion.[7] In their current state, networks will be slow to meet the demands of the new Internet of Things paradigm. Software Defined Networking is a new networking methodology that if applied properly can address these issues.

In this paper, we will cover the basics of SDN and examine an SDN architecture designed to work in multi-network IoT environments. First, we provide a view of a Software Defined Network and its components. Then, we present an example of a proposed SDN architecture that is designed to work in an IoT environment. The researchers responsible used a technique called Network Calculus to model the Quality of Service performance of the network. The Genetic Algorithm was adapted to optimize the performance of the architecture.

UMM CSci Senior Seminar Conference, April 2018 Morris, MN.

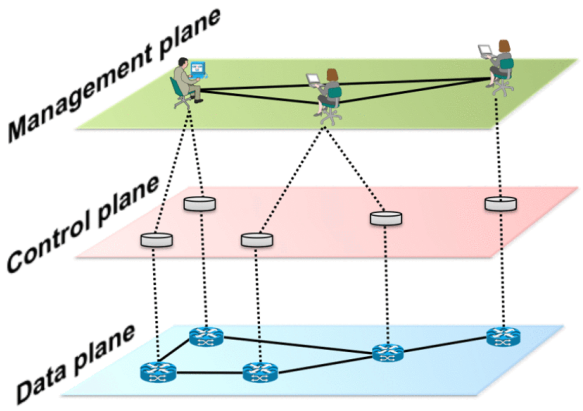

Figure 1: layered view of networking functionality [2]

2. BACKGROUND

2.1 Network Structure

A simplified view of networks can be organized into three planes. (see Figure 1) Forwarding devices such as routers and switches are on the bottom, in the data plane. A switch is a hardware device that is used to connect multiple devices on a local network. A router, which has a similar function to switches, is slightly more sophisticated. Routers transfers packets of data between different networks (for example, a home network and the internet). Packets are the formatted unit of data used in networking. They contain source and destination network addresses, as well as information that can be compiled into usable files. The control plane represents the decision making logic that governs the data plane. Above, the management plane is made up of the tools used to monitor and configure a network. Traditional IP networks are enormously complex and difficult to manage, as well as being near impossible to make significant changes to.[2]In a traditional network, the control and data plane

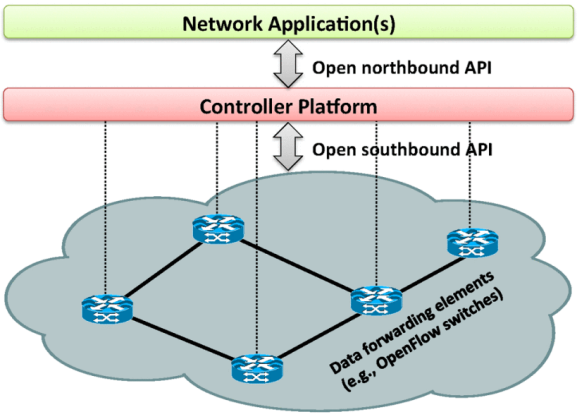

Figure 2: view of SDN infrastructure [2]

are vertically integrated. The control plane, which decides how to handle traffic, and the data plane, which forwards the network traffic, are bundled inside networking devices and highly decentralized. A network manager must individually configure each networking device, which greatly reduces the flexibility of the network and hinders innovation in networking technology. New network protocols can take years to to be designed, evaluated, and deployed. For example, the transition from IPv4 to IPv6 has taken roughly a decade and is largely incomplete. Changing network architecture is seen as unfeasible. Network configuration errors are extremely common. A single misconfigured device can cause a host of networking errors, like packet loss and forwarding loops, and sometimes a single misconfigured router can compromise the entire network.

2.2 SDN Infrastructure

The primary goal of Software Defined Networking is to separate the control and data planes. This is accomplished by creating a network operating system, or SDN controller. Instead of handling network decision making themselves, the forwarding devices become simply routers and switches. The SDN controller becomes the centralized "brain" of the network. The virtualization of these functions leads to several benefits. A view of the SDN infrastructure is given in Figure 2. We will begin by defining some of the terms used to describe the layers of an SDN network.

Forwarding Devices are data plane devices that perform actions on incoming packets of data. The actions forwarding devices take are defined by instructions from southbound interfaces. The Data Plane is made up of forwarding devices by wired or wireless connections. The Southbound Interface is between the control and data planes. The southbound API defines how forwarding devices and the control plane interact. The control plane is the intelligent logic that forms the "brain" of the network. Forwarding devices are programmed by the control plane through the southbound interface. The control plane houses all applications and controllers that govern the forwarding devices.

The Northbound Interface abstracts the lower-level functions of the southbound interface. It is used as an API for developers to send instructions to the forwarding devices. Network Applications provide monitoring and configuration of the network. Firewalls and load balancers are also included here.[2] This is equivalent to the Management Plane of the previous view.

With SDN, an administrator can change any network switches rules when necessary – prioritizing, deprioritizing or even blocking specific types of packets. it enables the administrator to manage traffic loads in a flexible and more efficient manner. A network administrator need only deal with one centralized controller to distribute policies to the connected switches, instead of configuring multiple individual devices. SDN also virtualizes hardware and services that were previously carried out by dedicated hardware, resulting in the touted benefits of a reduced hardware footprint and lower operational costs.

2.3 OpenFlow switches

OpenFlow is a communication protocol that allows network controllers to to manage the delivery of data packets over a network of switches. It was one of the first programmable networks developed, and has become synonymous with Software Defined Networking. OpenFlow switches are utilized in the SDN architecture studied in this paper. OpenFlow was developed by a team of researchers at Stanford University as a way to experiment with new network protocols and network traffic.[3]

OpenFlow switches work by providing an open protocol to program the flow table in different forwarding devices. An OpenFlow switch is made up of three parts: A Flow Table that tells the device how to process incoming data, a Secure Channel that connects the switch to an SDN controller and the OpenFlow Protocol, which is the standard for an OpenFlow-enabled controller to communicate with a switch. Switch and router manufacturers will be able to add OpenFlow functionality to their existing hardware by installing the Secure Channel software and implementing a Flow Table.

3. IOT SDN CONTROLLER

In order to address the changing network landscape and new IoT paradigm, researchers from the University of California Irvine have developed and tested a novel controller. This controller extends their previous work on MINA (Multi-network INformation Architecture), which is a middleware designed specifically to work with dynamic and heterogeneous networks. The QoS performance (delay, throughput, and jitter) was analyzed using the Network Calculus model. Throughput is defined as the rate of successful data delivery per second, similar to bandwidth. Delay, or latency, is how long it takes for a unit of data to travel through the network. In networking, jitter refers to the variance in sending data packets. If packets are not sent in equal intervals, it could result in congestion of the network [8]. The team developed a Genetic Algorithm based routing control algorithm in order to make the network more flexible and adaptable. This GA-based flow scheduling algorithm is compared to the common scheduling algorithms bin packing and load balancing.

3.1 MINA Middleware

MINA was designed as a platform that can operate in multi-network environments and maximize network capabilities. Its chief design philosophies are a tree-based hierarchical structure and and the Observe-Analyze-Adapt approach, as seen in Figure 3. At the top of a MINA system, Tier 1, is a centralized server that handles information sent from the de-

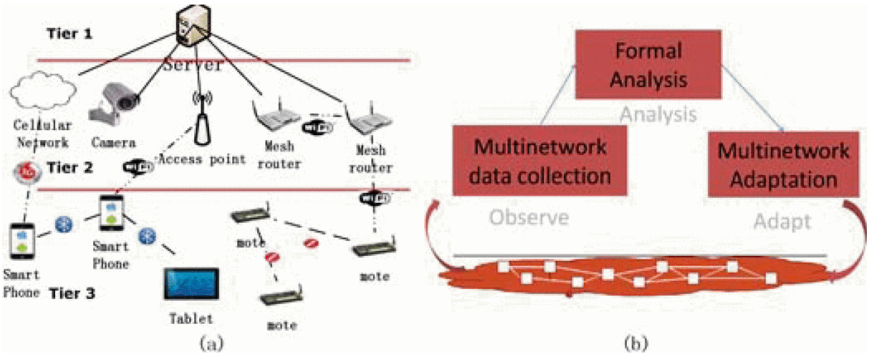

Figure 3: OAA Paradigm of MINA [5]

vices beneath it. Tier 2 consists of resource-handling nodes, such as routers and access points. Beneath that is Tier 3, which is made up of mobile nodes that are connected to the Tier 2 nodes.

The first step of the OAA approach is observe. The nodes in a network are evaluated and placed in a tree structure based on their capabilities and parent-child relationship with other nodes. Network analysis proceeds in the Analysis step. The formal method Maude is used to judge the network transmissions from the nodes and generate a set of QoS parameters. Using the information gathered in the previous steps, network resources are dynamically reallocated to maximize efficiency.[5]

3.2 Modeling with Network Calculus

In order to analyze the communication delays and performance of a communication network, a model called Network Calculus is used. Network Calculus is a methodology that provides insight into the packet sending and receiving problems in networking. Network calculus fundamentally relies on min-plus convolution.

The diagram in Figure 4 represents the transmission of bits of data on a node in a time interval [0,t). The arrival traffic, service capability, and departure traffic can be modeled as curves. A(t) represented the data volume that arrived, S(t) is the data volume served, and D(t) is the data volume departed. It is assumed that each node has a constant capacity R. The service curve is defined by $S(t) = R[t-T]^+$, where R is the capacity $[x]^+ = \max(0, x)$, and T is the transmission delay (the difference in time between the first bit entering the queue and the last bit leaving.[4]

The analysis of network traffic in Network Calculus depends on the technique of min-plus convolution, which is represented by the symbol $\otimes$. In mathematics, convolution is an operation on two functions that produces a third. Min-plus convolution is used to generate a departure curve from the arrival and service curves of data over time.

$$D(t) = A(t) \otimes S(t)$$

This is translated to:

$$D(t) \geq \inf_{s \leq t}(A(s) + S((t-s))$$

The operator inf stands for the infimum. The infimum, or greatest lower bound, of a subset X of a set Y is the greatest element in set Y that is less than or equal to all elements in

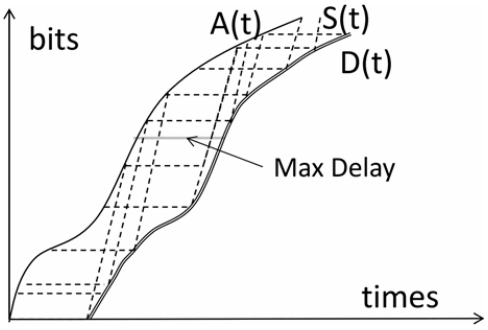

Figure 4: System with service, arrival, and departure curves

set X.[6] In this case, a point on the departure curve is found by adding the arrival volume of data and the transmission delay (S(t)).

Network Calculus is a useful measure if measuring Quality of Sevice (QoS) in a complex network for several reasons. First, the service curve of a node with multiple flows of data can still be calculated. It can be assumed that each node has a FIFO (first in, first out) scheduler. A flow i would have a leftover service curve:

$$S_i = \frac{\theta^i}{\sum_{j \neq i} \theta^j} R[t-T]^+$$

Here, R is the transmission rate of the node. θ is the weight, or data rate.

Sometimes data transferred in a network may make multiple hops to reach a destination. This is known as ad-hoc, or multi-hop networking. [1] Network calculus handles this by combining multiple service curves with min-plus convolution.

$$S(t) = S_1 \otimes S_2 \otimes ... \otimes S_n$$

A multi-hop path is demonstrated in Figure 5. The combined service curve S(t) is made up of the arrival and departure curves of several nodes. S(t) is made of several service curves chained together using the associative property of min-plus convolution. [4]

The Network Calculus model of the proposed SDN platform was tested against an experiment platform using the Qualnet network simulation software, as seen in Figure 6. The simulation was conducted on a two-hop network with two servers, one router, and 5 clients. One server provided

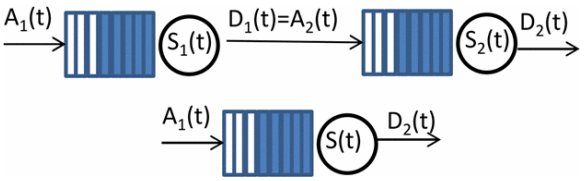

Figure 5: Association of service curve [4]

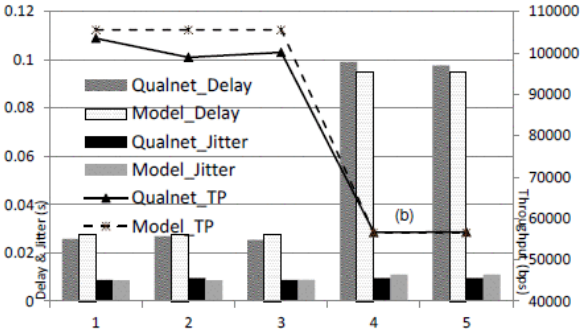

Figure 6: Validation results of Network Calculus method [4]

a video streaming service to the client, while the other a Skype audio service. Both servers used a 100Mbps Ethernet link to connect with the router. The clients connected to the server with a 2 Mbps 802.11b wireless link. [4] For each of the 5 clients in the graph, all three QoS parameters have a similar performance. The results of the simulation proved consistent with the Network Calculus model. This demonstrates that Network Calculus is an effective way to predict the performance of a network in an IoT environment.

3.3 Genetic Algorithm

An SDN controller can be used to optimize the performance of a network, and they are particularly useful in an Internet of Things network made up of data flows with differing levels of Quality of Service requirements. A centralized view of the network allows the SDN controller to precisely find the correct path for routing flows of data. In order to demonstrate the effectiveness of SDN in IoT environments , Qin et al implemented an SDN controller that uses a Genetic Algorithm based flow scheduling algorithm.[4]

The Genetic Algorithm is a method for solving optimization problems that is based on biological natural selection. Genetic Algorithms are used to find highly efficient solutions to search problems. The five general phases of the Genetic Algorithm are given in Figure 7, and are as follows:

1. The initial population is created. The population is made up of candidates. The candidates are possible answers to the search problem to be solved. Each candidate is made up of a set of variables called genes, tied together into chromosomes.

2. A fitness function determines the ability of each candidate, and assigns them a fitness score.

3. The selection phase picks the fittest candidates based on their scores. Two parents will be chosen from the candidates to pass on their genes to the next generation.

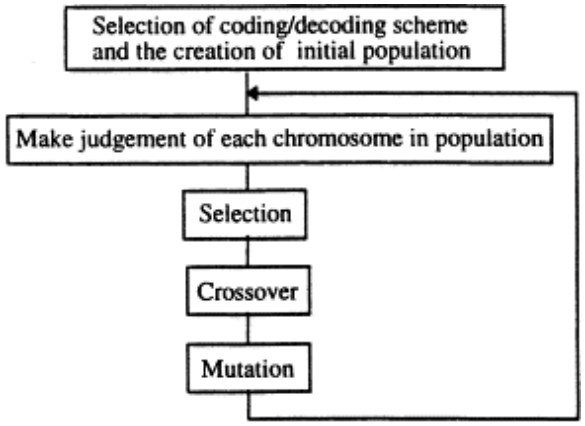

Figure 7: The general flow of GA [9]

4. Crossover involves exchanging the chromosomes of the parents to create offspring. The offspring are added to the population. The chromosomes of the offspring are made up of a combination of the parents' genes.

5. The final stage is mutation. Genes of the offspring's chromosome are changed randomly.

The population is kept at a fixed size, and as new offspring are created candidates with the lowest fitness scores are eliminated from the population. This is a process that repeats until the population does not create offspring that are significantly different from the initial population.[9]

This methodology was applied by Xiang et al to satisfy QoS requirements for networks. A network can be seen as a directed graph, where data is sent between each of the nodes (see Figure 8). The cost of traveling between two nodes can be weighted in terms of Quality of Service requirements. The communication path in a network can neatly match the chromosome concept of Genetic Algorithms. In terms of the Genetic Algorithm, each path through the graph from source node s to destination d is a member of the population. The chromosomes are made up of nodes along the path. No loops in the graph are allowed; meaning there are no repeated genes (or nodes) in the chromosome. A fitness value fro each chromosome can be calculated by adding the end-to-end delay, jitter, and throughput on the path.

When performing crossover, the top two ranked chromosomes with common genes are chosen as parents. Sub-paths of the parents' chromosomes are used to create offspring. During Mutation, a bottleneck node is chosen from the path where the node causes the most delay. This node is replaced by a random mutation path that can reach the same destination. For example, imagine two parent paths a,b,c,d,e and a,s,b,e. Two children are created, a,b,e and a,s,b,c,d,e from the crossover point b.

The output of the Selection and Mutation is eight chromosomes ranked according to fitness. The top two chromosomes become the parents of the next round. The algorithm runs until a chromosome with a suitable fitness score is generated or a predetermined generation size is reached.

4. METHOD

In order to test the performance of their SDN controller, Qin et al built a prototype in the Qualnet simulation plat-

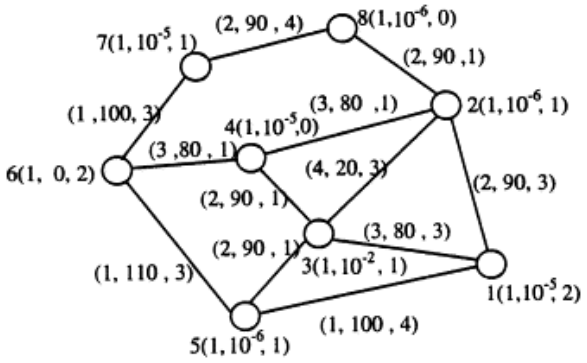

Figure 8: Topology of a network system, consisting of eight nodes and edges connecting the nodes. Each node is denoted by a tuple <ndl, lr, dv>, the elements being node delay, node loss rate, and node delay variation. Each edge is denoted by a tuple <cost, bw, and ldl>, the elements being cost, bandwidth, and link delay.[9]

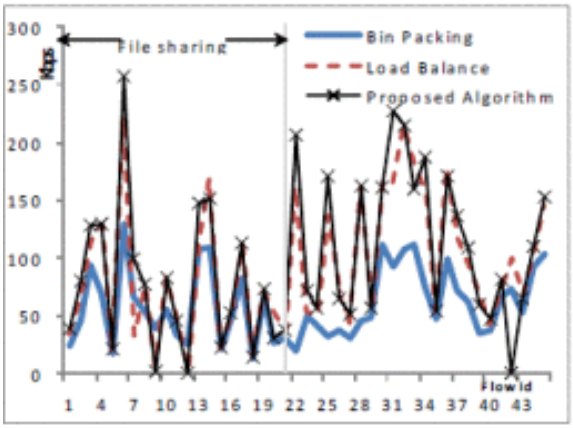

Figure 9: Throughput performance result [4]

form. The performance was compared against two other popular scheduling algorithms, bin packing and load balancing. The simulation was carried out over a network with three data servers, three edge switches, two core routers, and 12 access points with 45 end devices (there is one flow of data for each of the end devices represented on the x axis of Figures 9,10, and 11). The 45 flows are divided into the type of data they are transmitting. Flows 1-21 are file sharing, 22-36 are tele audio, and 37-45 are video streaming.

5. RESULTS

During the simulation, three QoS parameters were measured: throughput(Figure 9), delay(Figure 10), and jitter(Figure 11). In the throughput test, the proposed algorithm has an average 8% performance advantage compared to load balance. When testing delay,the proposed algorithm is significantly better for Tele-Audio flows than either of its competitors. For the jitter test, the proposed algorithm is on average better than Bin Packing or Load Balancing. The performance of the proposed algorithm compared to the others proves it is a feasible method of network flow scheduling.

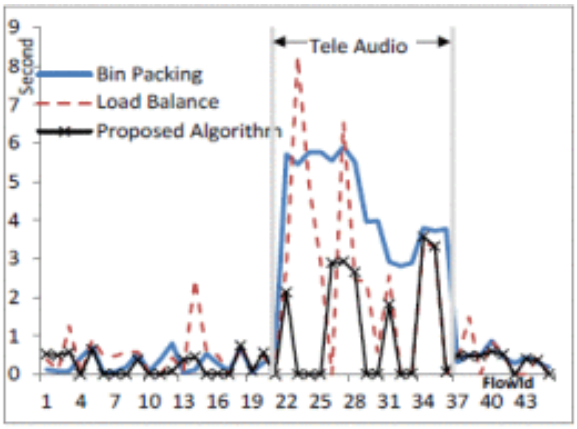

Figure 10: Delay performance result [4]

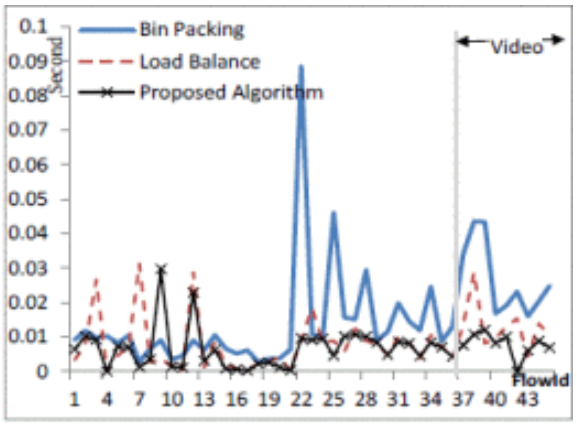

Figure 11: Jitter performance result [4]

6. CONCLUSION

Software Defined Networking is an emerging field in the space of computer networking. It attempts to address the issues of complexity and inflexibility that are present in traditional networks. Currently, forwarding devices on a network handle both the transmission of data and deciding how the data is handled. Software Defined Networking attempts to separate these two planes of activity. It does so by creating an SDN controller that manages the data forwarding activity of network devices. The centralized and programmable networks will lead to benefits like improved adaptability and control of networks.

In this paper, I examined a proposed SDN controller that is designed to be used in the IoT. The Network Calculus model was used to estimate the performance of the controller in a IoT multinetwork. The controller adapted the Genetic Algorithm to optimize network traffic, and was demonstrated to perform well against other algorithms. This controller could be an effective tool to manage network activity in a heterogeneous IoT multinetwork environment.

Acknowledgments

I would like to thank Kristin Lamberty and Elena Machkasova for their advising and feedback. I would also like to thank my alumni reviewer, Joseph Thelen, for taking the time to critique my paper.

7. REFERENCES

[1] R. W. Heath. Multi-hop networking.

[2] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, Jan 2015.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, Mar. 2008.

[4] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian. A software defined networking architecture for the internet-of-things. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–9, May 2014.

[5] Z. Qin, L. Iannario, C. Giannelli, P. Bellavista, G. Denker, and N. Venkatasubramanian. MINA: A reflective middleware for managing dynamic multinetwork environments. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–4, May 2014.

[6] Wikipedia contributors. Infimum and supremum — Wikipedia, the free encyclopedia, 2019. [Online; accessed 1-March-2019].

[7] Wikipedia contributors. Internet of things — Wikipedia, the free encyclopedia, 2019. [Online; accessed 1-March-2019].

[8] Wikipedia contributors. Network performance — Wikipedia, the free encyclopedia, 2019. [Online; accessed 1-March-2019].

[9] F. Xiang, L. Junzhou, W. Jieyi, and G. Guanqun. Qos routing based on genetic algorithm. *Computer Communications*, 22(15):1392 – 1399, 1999.