The Performance of Software Defined Networks Modeling and Simulation

[1]Mrs.K.Vijayalakshmi, [2]Mr.R.Kalaiselvan
[1]Lecturer, Computer Science and Engineering, Srinivasa Subbaraya Government Polytechnic college Puthur-609108,
[2]Assistant Professor, Computer Science and Engineering, Parisutham Institute of Technology and Science, Thanjavur -613006.
Email: mailtovijius@gmail.com

Abstract: **Software-defined networking** (**SDN**) technology is an approach to cloud computing that facilitates network management and enables programmatically efficient network configuration in order to improve network performance and monitoring. Software defined networking offers numerous benefits including on-demand provisioning, automated load balancing, streamlined physical infrastructure and the ability to scale network resources in lockstep with application and data needs. Software Defined Networking (SDN) has emerged as a new networking paradigm for managing different kinds of networks ranging from enterprise to home net-work through software enabled control. In this paper we enhance the Modeling a system involves the abstraction of its features and properties, focusing exclusively on those that are of interest to the study. As a result, a model can be understood as the logical representation of a system with different levels of complexity (normally less complex than the real system). Simulation is the imitation of a real-world system through a computational representation of its behavior according to the rules described previously in a model. When a system is simulated, it is mandatory to consider a limited number of characteristics, properties or behaviors of interest, so as to make the model tractable; otherwise, it will be infinitely more complex and detailed. We further identify some challenges and promising future directions on SDN security, compatibility and scalability issues that should be addressed in this field.

Keywords: Software Defined Networking, Network simulator, Modeling Techniques, Network Simulators

1. Introduction

Computer networks generally consist of a large number of network de-vices such as switches, routers, and middleboxes (i.e. devices which process traffic other than forwarding). A large number of servers and hosts are inter-connected through these network devices and middleboxes. These network devices and middleboxes are vendor specific and they have proprietary solutions. Network operators are responsible for configuring each device to handle network and security events. Configuring these devices manually with low-level device specific syntax is a tedious, complex, time consuming and error prone task, as network operators are required to be present all the time to configure these devices. This is a main reason for network downtime [1, 2], be-cause automated configuration and modification are absent in the traditional IP networks [3]. However, considering the current network dynamics, it is very important that a network can adapt to the current status automatically. But, in the traditional IP network it is difficult to configure the network dynamically according to the current status.

To make it even more difficult, in the traditional network, control and data plane are integrated inside the network devices, which may reduce flexibility and dynamicity. Moreover, the lack of programmability and centralized con-trol in the traditional network makes it difficult to deploy new services without halting the ongoing services [4]. The scenario could be made even worse be-cause of the enormous size of a network [5]. .

The recent emergence of Software Defined Networking (SDN) addresses these issues by decoupling the network control from the data plane of the net-work devices [6, 7]. In SDN, network intelligence is logically centralized in a software based entity called controller, and network devices like switches and routers behave as a simple forwarding device. Forwarding devices contain the flow rules to process the incoming packets based on match fields mentioned in the flow tables (such as source IP, destination IP, protocol, etc.). These forwarding devices can be programmed by the controller using a standard interface (such as OpenFlow [8], NETCONF [9], and Forces [10]). Moreover, network operators can define the high-level network policy at the controller, which can be enforced in the switches by the applications running on the controller.

In addition, as SDN controller can collect the flow statistics from the switches in the network, it can provide real-time global network status by analyzing flow statistics i.e., via the applications deployed at the controller. This can dramatically simplify the network architecture and provide effective network management.

Motivations. The global visibility and programmability of the SDN controller open up new avenues for network security. Recently, cyber-security has received more attention from academia as well as from industry According to the report of Symantec, cyber attacks continue to grow rapidly. Obviously, deployment of services in the network without adequate security may cause a lot of risks and vulnerabilities, which can be exploited by attackers. Moreover, security is a major concern not only in enterprise and ISP network, but also in different types of networks. To tackle the challenges posed by the growing cyber attacks, there is a need to dynamically configure and deploy the security policies on-demand. Currently, most network operators rely on statically deployed devices to protect the net-work from cyber attacks. It is a time consuming process and causes network operators to either over provision or under provision the resources.

Regarding SDN, Open Networking Foundation (ONF) that is an industry driven organization has been created by many service providers and different network vendors to help promote software-defined standards. SDN now has gained significant attraction from both academia and industry. Most of the vendors like IBM and Hewlett-Packard have already launched switching devices which support Open Flow protocol. Google has also deployed SDN to connect its data centers . On the other hand, academic researchers are also actively involved in the development and deployment of SDN. Computer-based models are usually classified as follows:

Deterministic vs. Stochastic: A deterministic model predicts a specific output from a given set of inputs with neither randomness nor probabilistic components. A given input will always produce the same output given the same initial conditions. In contrast, a stochastic model has some inputs with randomness, so the model predicts a set of possible outputs weighted by their likelihoods or probabilities [3].

Steady-state vs. Dynamic: A steady-state model tries to establish the outputs according to the given set of inputs when the system has reached steady-state equilibrium. In contrast, a dynamic model provides the system reactions facing variable inputs. Steady-state approaches are often used to provide a simplified preliminary model [4].

Discrete vs. Continuous values: A discrete model is represented by a finite co-domain; hence, the state variables take their values from a countable set of values. In contrast, a continuous model corresponds to an infinite co-domain. Therefore, the state variables can take any value within the range of two values [5]. However, there are some systems that need to be modeled showing aspects of both approaches which bring about combined discrete-continuous models [6].

Simulations can be carried out following two approaches: local and distributed. Distributed simulation is such that multiple systems are interconnected to work together, interacting with each other, to conduct the simulation. In contrast, a local simulation is carried out on a single computer. Historically, the latter approach has been the most widely used to simulate computer networks, but the increasing com-plexity of simulations is fostering the importance of the former approach [4].

Behavioral information extracted from a real system is used jointly with system specifications and requirements. Based on these inputs, a system model is built and subsequently validated through simulation. Usually, several performance metrics are determined during simulations, which can be compared with results extracted from experimentations with the real system. If both are similar, the model is considered to be valid, while if they are not, the system model must be corrected. Similarly, performance metrics are used to determine if the system model fulfills the requirements, so the designed system can be refined and extended in a controlled way.

2. Software-Defined Networking Architecture

Computer networks is mainly comprised of three planes: the forwarding, control and management planes. The forwarding plane consists of network-ing devices such as routers and switches which are responsible for forwarding traffic. The protocols used to deploy the rules in the forwarding tables of the data plane devices reside at the control plane. The management plane consists of the network and security policies to manage the network. Control plane enforces these policies in the forwarding plane. In the traditional computer networks, the control and data plane are tightly coupled with each other and integrated in the same networking devices. It makes the network rigid and static in nature.

Because of strong coupling between control and data plane in traditional computer networks, it becomes very tedious and complex task to develop and deploy new network applications

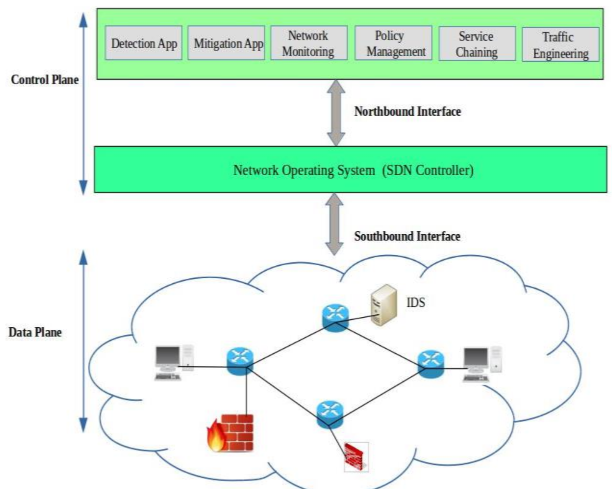


Fig. 1, the SDN architecture

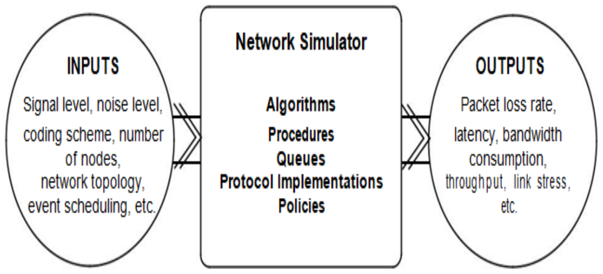
Software-Defined Networking (SDN) is a new networking paradigm which separates the control plane from the data plane. It is also comprised of the three planes: the data, control and management plane. To represent the work on SDN, the term was coined at Stanford University. Since then, it has at-tracted attention of researchers from both academia and industry. Fig. 1 illus-trates the SDN architecture. The separation of control and data plane enables to deploy network applications based on the current network requirements. Many novel network and security applications based-on SDN have been pro-posed. The main reason, for researchers to have interests in SDN is mainly because it provides centralized control where policies can be expressed for varying network conditions and these policies can be enforced in the data plane through southbound API. Moreover, SDN offers global visibility and programmability which allows configuring the networking devices automatically based on the current network status. The Southbound interface provides an interface for communication between the data plane devices and the control plane elements. It specifies the communication protocol between data plane devices and the SDN controller. The control plane configures the network devices in the data plane through southbound interface which offers the communication between the data and control plane. It is also called as a network operating system (NOS) or brain of the network. All the logic resides in the applications and the SDN controller, which together is called as the control plane. The Northbound interface offers an interface to developers for developing applications. It hides the low-level implementation details to pro-gram data plane devices.

Abstraction: The applications deployed at the controller are offered an ab-stract view of the network. Applications can specify the desired network behavior through high-level policy languages. These high-level configurations can be mapped into data plane configuration by the SDN controller.

Flow-based Management: Forwarding decisions in the SDN switches are taken on per flow basis. The basic characteristics of SDN is to forward the flow to the controller, when the network devices do not have the rules to handle these flows. This feature enables the network administrator to deploy the rules as and when it is required. It enables to deploy the network and security applications for on-demand services in the data centers and service provider network.

3. Computer Network Simulation

Nowadays, computer network simulation is a fundamental element in network researching, development and teaching. Network simulators are widely used not only by practitioners and researchers, but also by students to improve their under-standing of computer networks. Network simulation can be extremely useful when applied to scenarios such as protocol analysis, complex network deployment, eval-uation of new services, prototypes or architectures, and so on. Broadly speaking, a network simulator can be understood as a black box, as illustrated in Figure 3.1.


INPUTS
Signal level, noise level, coding scheme, number of nodes, network topology, event scheduling, etc.
Network Simulator
Algorithms
Procedures
Queues
Protocol Implementations
Policies
OUTPUTS
Packet loss rate, latency, bandwidth consumption, throughput, link stress, etc.

A network simulator implements a computer network model through algorithms, procedures and structures according to a given programming language. The implemented network model receives a set of parameters as inputs, which impose the rules and constraints of the simulation. In other words, these parameters set the course of the simulation. When the simulation process ends, the network simulator returns a set of results as outputs that can be used with a dual purpose: validating and verifying the correctness of the implemented model for a particular domain of system states, and analyzing the behavior of the network modeled when the former has been successful.

Computer networks are most often simulated with discrete-event simulation. The main reason behind this widespread use is that discrete-event simulation adapts better to represent the behavior of computer networks, since computer net-work protocols can be modeled as finite state machines. In a computer network there is a steady state between two consecutive events, and discrete-event simulation allows for jumping from one steady state to another, leading to faster simulations. Other interesting aspects of discrete-event simulation are flexibility and lower computational overhead.

4. Performance Modeling

As pointed out previously, modeling is the first step in the evaluation of the sys-tem performance, where the system is abstracted through a properly detailed model that reproduces the system behavior under the environment influence (workload). In the most specific performance evaluation studies, the main objective is to obtain a set of performance metrics such as throughput and response times in the steady-state of the system. First, modeling techniques for performance evaluation are introduced in this section. Then, the key aspects to consider when modeling protocol layers and workloads are outlined. Finally, some possibilities to model the network topology are presented.

4.1 **Modeling Techniques**

The most common techniques for performance modeling of discrete-event dynamic systems in general and computer networks in particular are Markov Chains, Queuing Networks and Petri Nets [5]. Markov Chains provide a general framework to model discrete-event dynamic systems, while Queuing Networks and Petri Nets allow for building high-level models that can also be translated to Markov Chains. These high-level models are closer to the structure of the real systems and so it is easy to match system and model components, whereas Markov Chains put the emphasis on describing the behavior of the system at the state space level. All these techniques use stochastic processes for representing times as the way of properly analyze the performance of the modeled systems.

Markov Chains are stochastic models described by graphs, where nodes corre-spond to system states and transition between states are represented by links. The probability associated to each transition is also annotated in the graphs.

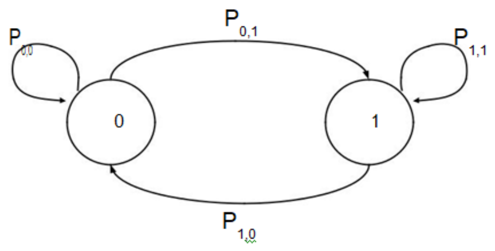


Figure 4.1 shows an example of a Markov Chain model with only two states and the corresponding probabilities of changing the state or staying in the same. An important property of Markov Chains is that the future states only depend on the present state, but not on the previous states that is, they are memory less. Thanks to this property, Markov Chains have the great advantage of low analysis complexity. Continuous Time Markov Chains (CTMC) are the Markov Chains commonly used to model discrete-event dynamic systems.

When the systems are complex, the number of possible states is high, and modeling them directly with CTMC becomes difficult. In these cases, more abstract or high-level models like Queuing Networks and Petri Nets are better alternatives.

5. **Performance Metrics in Computer Network Simulation**

After being modeled using some of the techniques previously presented, the performance of a computer network can be evaluated at different layers. To extract accurate information from a simulation, it is mandatory to select the proper metrics according to the studied layer. In this section, the most widely used metrics to evaluate network performance are summarized. These metrics are related to the TCP/IP stack. They are also classified according to their usefulness to wired and wireless networks.

5.1 **Link Layer**

A metric commonly used to determine throughput at this layer is the ratio of bits received to the duration of the transmission. Another important throughput metric is the nominal channel capacity (NCC), which is the maximum number of bits that can be transmitted per time unit. NCC is subsequently used to calculate the effective channel capacity (ECC) by not considering the overhead introduced by the protocols used in the communication. Throughput can also be estimated using the channel utilization (CU), which is the ratio between NCC and the number of bits received per transmission time [14].

Transmission errors are also important at this layer. There are two main metrics to evaluate transmission errors: bit-error rate (BER) and packet-error rate (PER). The former represents the ratio of the number of received bits that have been altered while traversing the communication channel to the number of sent bits, while the latter indicates the ratio of the number of incorrectly received data packets to the number of received packets. In both cases, the higher the metric value, the worse the network performance.

The aforementioned metrics can be applied both to wired and wireless net-works. However, there are metrics to determine performance exclusively related to the latter. For example, the spectral efficiency, which is the number of received bits per unit time per unit bandwidth and per unit area (b/s)/(Hz 3 m^2) [15].

There are other metrics that can be used to estimate other network perfor-mance aspects at link layer in wireless networks. The received signal strength indication (RSSI) represents the signal strength observed at the receiver’s antenna during packet reception [16]. The access point transition time (APTT) evaluates the duration of the handover [17]. The Jain’s index evaluates fairness when multi-ple nodes are attempting to access the wireless medium [18]. Furthermore, there are metrics to evaluate the quality of the signal both in wired and wireless networks. The signal-to-noise ratio (SNR) metric is used in wired networks to calculate the ratio of the signal power to the background noise. In wireless networks the analogous metric is the signal-to-interference-plus-noise ratio (SINR), where the interference power of other signals is also considered.

5.2 Internet Layer

This subsection focuses on describing the metrics used to estimate the performance of two main routing tasks: path selection and network topology management. First, the path selection process of a routing protocol tries to determine the best path to a destination node, using several metrics to evaluate the cost of the path between source and destination. In wired networks, the most widely used metric is hop count, which helps find the minimum hop-count routing between source and destination. Other metrics related to path selection in wired networks are the available bandwidth of the link and the round trip time (RTT), which measures the round trip delay of unicast probes between neighboring nodes.

On the other hand, although the aforementioned metrics can also be used in wireless networks, other metrics are specifically applied in practice. For example, path selection metrics exclusively used in wireless networks are the expected transmission count (ETX) and the expected transmission time (ETT). The former predicts the number of retransmissions needed to send a data packet over a link, while the latter determines the time a data packet needs to be correctly transmitted over a link. The expected transmission time was extended considering channel diversity to obtain a trade-off between delay and throughput in a path selection metric called weighted cumulative expected transmission time (WCETT). A network topology imposes how network entities are interconnected with each other and hence establishes how data is forwarded through the network. Communications developed on higher layers depend on the organization and management of the underlying network, so a specific network topology affects the routing scheme, the scalability and the complexity of the network. Metrics commonly used to estimate the correctness of a network topology are betweenness and node degree. The former is the number of shortest paths between any two nodes that go through a particular node, quantifying the importance of such a node in the information exchange. The latter determines the number of nodes that depend on a specific node.

6. Discrete-Event Simulation

Discrete-event simulation deals with system models in which changes happen at discrete instants in time (events), rather than continuously, and state variables do not change in the intervals between these discrete instants.

Computer networks are most often simulated with discrete-event models, and there are two basic types of discrete-event simulation: trace-driven simulation and stochastic simulation. In trace-driven simulation, the simulation inputs come from data captured on the real system (traces), so the accuracy is maximized, but is not applicable to all types of systems. In stochastic simulation, the system work-load is characterized by probability distributions, using random values as inputs to the simulation model. This section first introduces random value generation. Then, the three simulation approaches for a stochastic simulation program (event-driven, process-oriented and parallel) are presented.

6.1 **Random Value Generation**

To develop simulations properly, it is mandatory to generate random values with predefined statistical distributions [6]. Generally, this is accomplished in two steps, which are briefly introduced in the following paragraphs. First, a step called random-number generation, in which a sequence of random numbers between 0 and 1 is generated with a uniform distribution.

Second, a step called random-variate generation, in which the previous sequence is used to generate a new sequence with the desired distribution

6.2 Event-Driven Simulation

In event-driven simulation, system state changes are caused by events, and the changes are considered instantaneous. Two facts are necessary for the simulation to evolve: when the events take place and how they change the system state. At the heart of the event-driven simulation is the so-called event list, a time-sorted list of the current planned events for the system. The first event on the list (at the head) is the closest in the future. event identity, event time, event type and event info. The simulation also maintains a clock with the time corresponding to the event being currently pro-cessed. The type field is used to select the event service routine to be executed, which also uses the info field during execution. As a result of the execution, the system state variables are changed and possible new events are added to the event list.

6.2.1 Simulation performance and scalability

PDES can improve the simulation both in terms of execution speed (performance) and network size (scalability) depending on the adopted parallel strategy. Time Parallel Simulation only achieves performance improvement, whereas Space Parallel Simulation can improve both performance and scalability. In some specific cases, both Time and Space Parallel Simulation can be used, further improving the execution speed. These strategies will be detailed below. An additional nonparallel and trivial strategy to reduce the execution time of a set of simulation experiments consists of concurrently executing different simulations on different machines.

The performance of a packet-level network simulator, where simulation involves packet processing and transmission from source to destination nodes through routers, can be characterized by the Simulated packet transmissions per second (PTS). This metric allows for estimating simulation times when the packet traffic and the average number of intermediate nodes (routers) are known:

$$T\ 5N_{pt}=PTS \text{ and}$$

$$N_{pt}\ 5\ N_f\ \ P_f\ \ H_f$$

In the equations above, T is the execution time of the simulation, N_{pt} the num-ber of packet transmissions, N_f the number of flows, P_f the average number of

7. Validation and Verification

Simulation models must be validated and verified before starting the analysis of simulation results and taking decisions based on the analysis. The validation of a simulation model is the process to assure that the design and assumptions taken are reasonable, and therefore, the model will provide results consistent with those of the real system. The verification of a simulation model is the pro-cess to check the correctness of the implementation of the design and the assumptions in a simulation program. A simulation model can be in four possible states:

- invalid and unverified
- invalid and verified
- valid and unverified
- valid and verified

When the modeling of a system and the programming of the model are developed by different teams, the modeling team is responsible for validation and the programming team is responsible for verification.

8. Network Simulators

The discrete-event simulation is the most widely used approach to study the behavior of a network. The main reason for using this approach is that it allows for obviating all those system states between two discrete points in time. Discrete-event simulation makes it possible to simplify a complex real process into a set of primitive actions. Thus, each primitive corresponds to a state change, and is represented by an event in the simulation. For example, the process of sending a data packet from a source to a destination is composed of several complicated tasks related to encoding and decoding data, signal transmission and data processing and verification. In contrast, under the discrete-event approach the aforementioned process could be depicted by only two events: 1) when the source has sent the packet, and 2) when the destination has received the packet, but a higher number of events might be defined for a deeper and more exhaustive analysis of the process.

A network simulator based on the discrete-event approach has two mandatory structures: a simulation time variable and a list of pending future events. The for-mer represents the time at which the current state of the system is known and represented in the simulation environment. The latter is a list containing\ state changes that have been scheduled to occur in the future, which rule the course of the simulation. Each network simulator has an API and a programming lan-guage that encompass these mandatory structures. Additionally, more specialized structures can be implemented such as a scheduler entity to manage the pending list by adding and removing events.

The simulator keeps track of a list of pending future events that have been scheduled to be executed at a specific simulation time. The simulator executes the events in sequential increasing time order. Specifically, the simulation time immediately jumps forward from the scheduled execution time of an executed event, to the execution time of the next event. Furthermore, once the execution of an event ends, the simulator checks the list of pending events. Then, the simulator will move to the next event, or will terminate the simulation if there are no more events in the pending list. It should be noted that an event execution may imply the emergence and the scheduling of one or more additional future events. Before a simulation takes place, several processes may be developed in the network simulator such as a network topology definition, and a link and node configuration. When the simulation ends, the performance can be determined through the analysis of the data available on time-stamped traces returned by the simulator.

9. Simulation

One of the main reasons for using simulation to assess the performance of the overlay is the impossibility of requesting a high number of users to participate in synchronous e-training activities for testing. This would be extremely expensive and disruptive for users. Besides, the network resources required to evaluate the performance of the overlay thoroughly might not be available or may be too expensive.

The ns-3 simulator is used to simulate the operation of the network of an organization geographically dispersed in several sites, the entities of the overlay and the participants within a synchronous e-training activity. The ns-3 simulator includes a vast number of models of wired and wireless channels and many of the elements that can be found in modern networks. It is written in C11 and uses the combined multiple-recursive random number generator MRG32K3a proposed in [19]. The whole TCP/IP stack is implemented on top of the channel models, so programmers can reuse the algorithms (even the code) of their implementations to build their ns-3 models. Thus, existing RTP and SIP libraries are integrated in the simulator by using the socket library of the simulator instead of the socket library of the operating system in order to simulate the overlay.

The model assumes that there exist various sites where IP multicast is available and there is an RTP relay in each site. An ideal wide area network (WAN) connects every site through 20 Mbps network links with a delay of 20 ms. The data rate of the network link connecting the RP to the WAN is 10 Mbps and introduces no delay in communications. These parameters are obtained empirically from the cor-porate network of a large corporation [18]. A CSMA channel is used to simulate the WAN between the sites. This channel is modeled as a simplistic

Ethernet-like network in which the state of the medium is instantaneously shared among all the devices, so there is no need for collision detection. A queuing approach is used in this model. The local area network (LAN) of each site is also modelled as a CSMA channel. A point-to-point channel, modeled as a simplistic point-to-point serial line link, interconnects each LAN with the ideal WAN. Both the CSMA and point-to-point channels model both the physical and link layers of the protocol stack.

The range of possible overlay deployments is extremely high. Many parameters such as the number of relays in the mesh, the number of participants and the balance of participants between relays can be varied. However, in order to evaluate the impact of each parameter individually, only one is varied at a time during the simulations. For simplicity, only balanced deployments are considered—that is, all the relays serve the same number of participants.

A participant has to be modeled to simulate the traffic generated during a synchronous e-training activity. The participant can use several kinds of media types to communicate with other participants in an activity. In this case, the participant is represented as an entity that generates one audio stream, one video stream, annotations on the shared whiteboard and telepointer movements, resembling the data generated by a synchronous e-training tool [18].

The key decision is how to model the traffic generated by a participant. Traces reporting the behavior of users can be collected from real e-training activities, so the model of a participant can be derived. For example, a participant may join the activity after a waiting time, simulated using an exponential random vari-able Exp(1/45) s. Then, the participant establishes an SIP dialog with the RP to join the activity and negotiate the media configuration. The audio and video streams of a participant are activated regularly. A video stream is a simulated 160 3 120 variable bitrate H.264 video stream at 10 frames per second (the size of the video packets sent to the network also has to be modeled). An audio stream is a simulated constant bitrate iLBC audio stream with a packetization time of 20 ms. Both the interval between activations and the duration of the streams are assumed to be normal random variables. Thus, a participant generates an audio stream of duration N(15, 2.6) s after an elapsed time N(25, 3.3) min, and gener-ates a video stream of duration N(30, 9.5) s after an elapsed time N(40, 8.2) min.

10. Analysis of Results

Different performance metrics can be used to analyze the overall performance of the overlay. The workload at the relays when forwarding streams can be esti-mated using the average bandwidth consumption. The user experience when participating in synchronous e-training activities supported by the platform can be estimated using the average packet loss and the interarrival jitter of the audio and video packets.

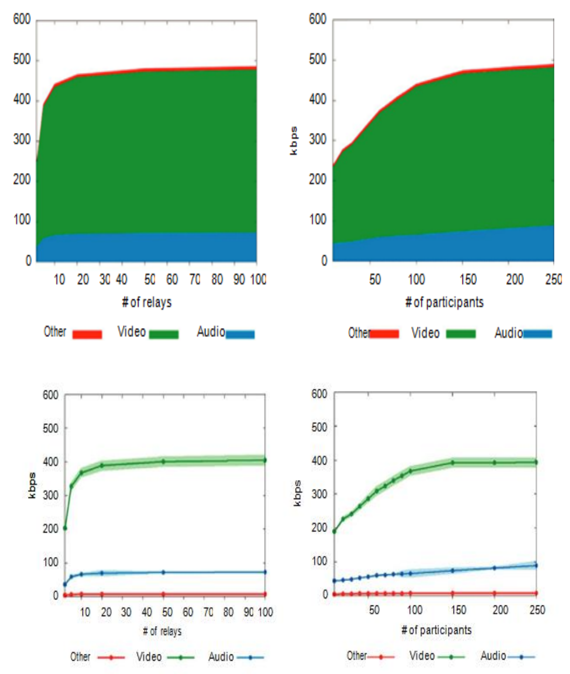


Figure:10.1 Average network bandwidth consumption in the network links of the sites for each media type.

This proves that the network bandwidth consumed in the links of the sites grows asymptotically due to the floor control policy limiting the maximum number of active streams in the overlay. Since stochastic processes modeling different entities of the overlay are involved in the simulations, confidence intervals are very useful to describe the accuracy of the results.

11. Summary

Modeling and simulation (M&S) are attractive and widely used techniques for the study of the performance of computer networks. They provide detailed results without disturbing network operation or even without the need of network avail-ability. This chapter summarizes the whole topic of performance M&S applied to computer networks. After an introduction, the main modeling techniques and per-formance metrics used in computer networks have been presented. Computer net-works are discrete-event systems and so discrete-event simulation has been highlighted as the most common simulation technique for executing computer network models. Random number generation, event-driven or process-based simulation, and parallel discrete-event simulation are important techniques related to discrete-event simulation covered in the chapter. Confidence in simulation results is only possible if the models are validated against the real systems and the simulation algorithms are verified against the models, so conditions for both have been detailed. The architecture of a simulator and the characteristics of the most used simulators complete this high-level panoramic overview of the performance M&S of computer networks. The chapter finishes with a case study of performance M&S for an application layer overlay network. In this paper, we surveyed existing research regarding the application of SDN on securing computer network. Relevant research in SDN community is still in its early stages, but there is immense work has been done to design and develop different applications to ease the burden of network management by means of SDN. We categorized existing work into seven categories: attack detection, attack mitigation, dynamic configuration based on SDN, Security policy management using SDN, traffic engineering achieved using SDN, deployment of middleboxes, and service chaining.

References

[1] Open Networking Foundation, SDN Security Considerations in the Data Center, Tech. rep., ONF (2012).

[2] Open Networking Foundation, SDN Security Considerations in the Data Center, Tech. rep., ONF (2013).

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

[4] R. Enns, M. Bjorklund, A. Bierman, J. Schönwälder, Network Config-uration Protocol (NETCONF), RFC 6241 (Jun. 2011). doi:10.17487/ RFC6241.

[5] J. M. Halpern, R. HAAS, A. Doria, L. Dong, W. Wang, H. M. Khos-ravi, J. H. Salim, R. Gopal, Forwarding and Control Element Separation (ForCES) Protocol Specification, RFC 5810 (Mar. 2010). doi:10.17487/ RFC5810.

[6] A. Tootoonchian, M. Ghobadi, Y. Ganjali, OpenTM: Traffic Matrix Esti-mator for OpenFlow Networks, Springer Berlin Heidelberg, Berlin, Hei-delberg, 2010, pp. 201–210. doi:10.1007/978-3-642-12334-4_21.

[7] N. L. M. van Adrichem, C. Doerr, F. A. Kuipers, Opennetmon: Network monitoring in openflow software-defined networks, in: 2014 IEEE.

[8] Garzia RF, Garzia MR, editors. Network modeling, simulation, and analysis. Electrical and computer engineering. Taylor & Francis; 1990.

[9] Guizani M, Rayes A, Khan B, Al-Fuqaha A. Network modeling and simulation: a practical perspective. Hoboken, NJ: John Wiley & Sons; 2010.
[10] Taylor HE, Karlin S. An introduction to stochastic modeling. 3rd ed. San Diego, CA: Academic Press; 1998.

[11] Burbank J, Kasch W, Ward J. An introduction to network modeling and simulation for the practicing engineer. The ComSoc guides to communications technologies. Hoboken, NJ: John Wiley & Sons; 2011.

[12] Wainer GA. Discrete-event modeling and simulation: a practitioner’s approach. Boca Raton, FL, USA: CRC Press, Inc.; 2009.

[13] Law AM, Kelton DM. Simulation modeling and analysis. New York, NY: McGraw-Hill Higher Education; 1999.

[14] Pidd M. Computer simulation in management science. Hoboken, NJ: John Wiley & Sons; 1988.

[15] Sokolowski J, Banks C. Principles of modeling and simulation: a multidisciplinary approach. Hoboken, NJ: John Wiley & Sons; 2009.

[16] Marsan MA, Bobbio A, Donatelli S. Petri nets in performance analysis: an introduction. In: Reisig W, Rozenberg G, editors. Petri nets. Vol. 1491 of lecture notes in computer science. Springer; 1996. p. 211 56.

[17] Bolch G, Greiner S, de Meer H, Trivedi KS. Queueing networks and Markov chains—modeling and performance evaluation with computer science applications. Hoboken, NJ: John Wiley & Sons; 2006.

[18] Wa¨hlisch M. Modeling the network topology. Springer; 2010. p. 471 86 [chapter 22].

[19] Sasnauskas R, Weinga¨rtner E. Modeling transport layer protocols. Springer; 2010. p. 389 99 [chapter 17].

[20] R. J. Colville, G. Spafford, ConfigurationManagement for Virtual and Cloud Infrastructures, Tech. rep., Gartner Inc (2010).