MiniNAM: A Network Animator for Visualizing Real-Time Packet Flows in Mininet

Ahmed Khalid, Jason J. Quinlan and Cormac J. Sreenan
Department of Computer Science, University College Cork, Ireland
Email: a.khalid@cs.ucc.ie, j.quinlan@cs.ucc.ie, cjs@cs.ucc.ie

***Abstract*—In this demonstration we present MiniNAM, a utility that provides real-time animation of networks created by the Mininet emulator. Mininet is one of the most well-known network emulators in research and academia. Although Mininet is capable of emulating both traditional and software-defined networks, it does not provide a tool to visually observe and monitor the packets flowing over the created network topology. Our utility includes all the components required to initiate, visualize and modify Mininet network flows in real-time. MiniNAM provides a graphical user interface that allows dynamic modification of preferences and packet filters: a user can view selective flows with options to color code packets based on packet type and/or source node. This establishes MiniNAM as a very powerful tool for debugging network protocols or teaching, learning and understanding network concepts. This demonstration illustrates a number of sample use cases and examples of using MiniNAM to create networks and view the generated network flows with customized preferences.**

***Index Terms*—Mininet, MiniNAM, Network Animation, Network Visualization, Software-Defined Networks, SDN**

I. Introduction

Network simulation tools such as NS2 [1] and emulation tools such as Mininet [2] are widely used for the validation of new network protocols in research and learning/understanding network concepts in academia. One of the main challenges with simulators and emulators is how to monitor the state of the network across what could be a large number of network entities and to analyze the complex and frequent message exchanges between these entities.

Packet traces generated by simulators contain static outputs and a huge amount of detail, limiting the user's ability to comprehend the data. Similarly in emulated networks, statistics are generally gathered in raw form or at the end of the emulation, hiding the dynamics of the protocol behaviour. Network visualization and animation tools address such problems. A network animator, like NAM [3] for NS2, allows users to quickly gather large amounts of traffic details, to visually identify patterns in communication, and to better understand causality and interaction.

Mininet is one of the most widely used network emulators. It creates a virtual network with fully operational nodes and allows interaction among these nodes, giving a more realistic and complete interpretation of a network in comparison to simulators like NS2. One of the key features of Mininet is its support for OpenFlow and Software-Defined Network (SDN) systems. Creating SDNs with Mininet can be as easy as running a single command. Mininet provides some visual tools like MiniEdit and Virtual Network Description (VND) [4] that further simplify the generation of network topology however, unlike NS2, Mininet does not provide a tool for visualization of traffic and packets flowing over the created network topology.

II. Design Overview

In this paper, we present MiniNAM, A Network Animator for Visualizing Real-Time Packet Flows in Mininet. MiniNAM is a graphical user interface (GUI) tool written with Tkinter and Mininets Python API (Figure 1). MiniNAM provides visualization of the main topology view, from which the user can derive a number of specialized views based on packet types and source or destination nodes. The UI provides a means of viewing traffic flows, monitoring traffic patterns as well as the packet-level details necessary to design new protocols. MiniNAM has no additional dependencies and can run on any system that has Mininet installed on it. MiniNAM contains all the components needed to create a network, to capture traffic flows, customize visualization parameters and finally display the run-time progression of packets from source nodes to destination nodes over appropriate links. While the scalability of MiniNAM is dependent on the hardware on which it is being run, MiniNAM can conveniently support networks of any size that can be created by Mininet on a certain machine. In our calculations, visualisation of the underlying Mininet topologies in MiniNAM generates little demand in overall CPU usage, but dependent on preferences and filters chosen, defined in later sections, overall CPU loading does increase but not significantly.

Like Mininet, to create a network, users can execute the MiniNAM utility from the command line by passing various arguments. To simplify the use of MiniNAM, arguments are kept in the same format and structure as the Mininet *'mn'* utility. To provide support for custom defined topologies, MiniNAM provides a *'–custom'* argument which can be used to load a user-defined python script. MiniNAM also supports

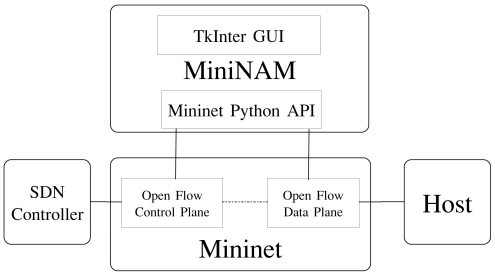
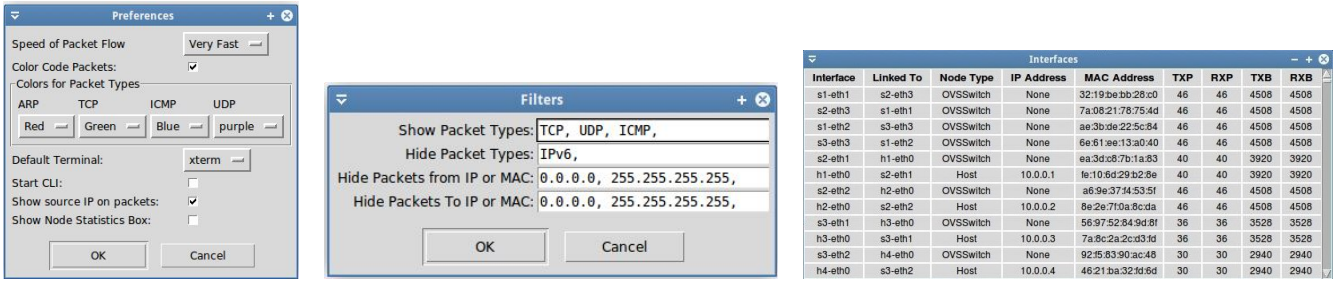


Fig. 1. System Design of MiniNAM



(a) Preferences (b) Filters (c) Statistics on all network interfaces

Fig. 2. Example of some of the dialog boxes available in MiniNAM.

TABLE I
PREFERENCES AND FILTER OPTIONS TO CUSTOMIZE FLOW DISPLAY

Preferences	
Speed of packet flows	Flows can be viewed in real time or adjusted to a slower speed
Color code packets	If selected, flows from each source node will have a different color
Colors for packet types	Option to choose a different color for different packet types
Show source IP on packets	If selected, first and last octet of IP address of source node will be displayed on every packet
Show node statistics	If selected, node statistics will be shown instantly on mouse hover over any node
Filters	
Show Packets Types	Specified packet/frame types will be displayed.
Hide Packet Types	Specified packet/frame types not displayed.
Hide Packets from IP/MAC	Specified source IP/MAC not shown
Hide Packets to IP/MAC	Specified destination IP/MAC not shown

networks with multiple controllers to enable visualization of more complex networks. To create such a network, a custom python script can be written and switches can be attached to one or more controllers. On execution of a command, a GUI will load and present the created topology with option to drag nodes around in the created window. (X,Y) positions can also be defined for the nodes in the python script.

A number of menus have been added to ease customization of the displayed flows:

- *Preferences:* As illustrated in Figure 2a, this menu, selected from the *Edit* menu of MiniNAM, permits a means of customizing the view of flows that will be generated. Options in the *File* menu can be used to save these preferences and load them later for other emulations.
- *Filters:* As shown in Figure 2b, this menu offer a means of limiting the types of packet being visualised, based on packet types, and respective IP and MAC addresses.

Each of the respective parameters in the menus above are further detailed in Table I. Further customization can be done by modifying the underlying source code. MiniNAM also provides other features through the *Run* menu that include: pausing the display of flows at a certain time, clearing the existing flows and viewing OpenFlow switch configurations. Once the traffic starts flowing, MiniNAM provides real-time information of the number of packets transmitted and received by every network node. Once this option has been turned on from the preferences menu, statistics will be shown when the cursor hovers over a node. In addition, statistical information of every interface in the network can be seen (Figure 2c) in the *Run* menu.

To provide interaction between elements in the network, each simulated node contains an option to open a terminal window by which additional network traffic or application level programs can be executed. All generated traffic will be displayed according to user preferences. The user can also observe the behaviour of network failures on the emulated network, by using the link down option to simulate a broken link.

III. APPLICATION OF MININAM

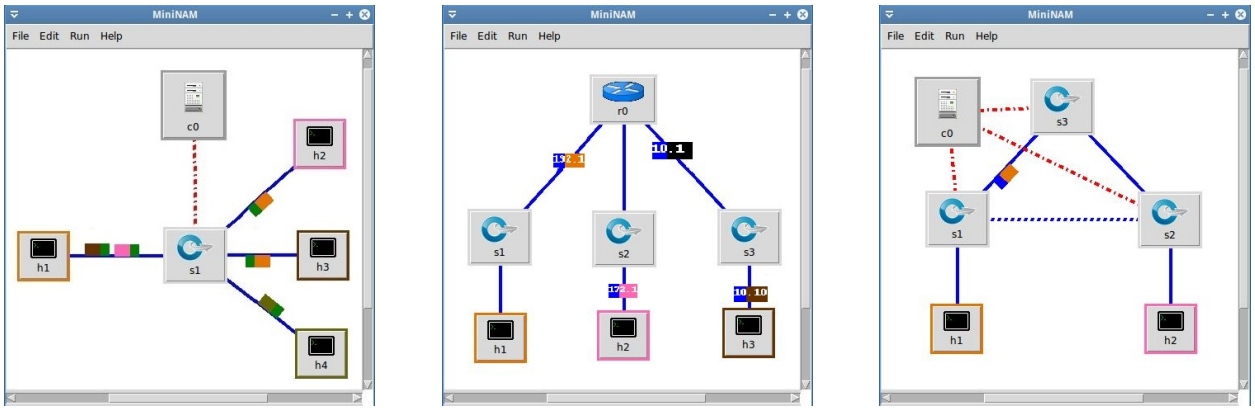
In this section we briefly discuss the role of MiniNAM in education and research.

- *Education:* Teachers can use MiniNAM to animate networking principles in class or within a lab scenario. Students can use MiniNAM to compare and understand network protocols. By slowing down the speed of network flows, and viewing how packets traverse the network, students can understand the relationship between the control and data aspects of the network.
- *Research:* Researchers can use MiniNAM to investigate new networking concepts, debug real-life network applications, as illustrated in [5] where an early prototype of MiniNAM was used to show real-time video streaming, and do more advanced comparisons by using the network interface summary to check the number of packets and/or bytes crossing each network node and interface.

IV. DEMONSTRATION OVERVIEW

In this demonstration we present our utility MiniNAM, through the use of three pre-defined distinct network protocol examples, and an interactive real-time UDP/TCP demonstration using iperf. We illustrate how MiniNAM can be utilized to visualize a fully functioning network and associated packet flows. Also, and more importantly, we show the effectiveness of MiniNAM when the control of the network inefficiently manages the network, through the incorrect implementation of either protocols, nodes or flows.

While the three pre-defined examples present a broad range of network protocols, the interactive demonstration illustrates



(a) Load balancing. The traffic from three servers can be viewed as different colors traveling towards the client.

(b) ICMP Packet visualization in NAT. The change in IP address of packets can be seen once a packet crosses router r0

(c) Re-routing packets. A broken data link (shown in dotted) results in packets following a different route (s1-s2-s3).

Fig. 3. Three network protocol examples illustrating the generated topology and associated packet flows in MiniNAM.

how to visualize and customized various parameters in MiniNAM. In this demonstration, we also initiate Mininet network topologies, customize preferences, generate traffic and visualize packet flows over the network. The three network protocol examples are:

- *Load Balancing:* This example uses a Ryu SDN controller to implement Server Load Balancing using an OpenFlow switch. A client requests traffic that can be served from three servers. The switch divides the traffic load among servers based on the logic implemented by the controller. To set up the example follow the steps at [1] to setup the load balancer and then run MiniNAM. This will create a topology with one switch and four hosts, as seen in Figure 3a. An echo server can be started on three of these hosts and if pings are sent from client, the packets can be seen distributed among the servers. This example illustrates an excellent use case where MiniNAM can make debugging easier when designing a new network protocol or algorithm.
- *NAT:* This example uses the simple linuxrouter example provided in Mininet to create a network with one NAT-enabled router connected to three switches. Three hosts, each on a different network communicate through a NAT enabled-Linux Router and if NAT is implemented properly, packets will be routed correctly and this can be seen in MiniNAM. MiniNAM can display the flowing packets as well as the change in IP address of packets when they pass through the router. Figure 3b outlines how the real-time visualization of packets can make it easier for students to grasp network concepts.
- *Re-Routing:* Taken from the Spanning Tree example in the Ryu controller, this example creates a network with multiple paths between hosts. If a path is broken and an alternate path is available, the controller tries to update the path. In this example Hosts are connected to each other through multiple paths. By simulating a broken link, the behavior of the routing protocol can be monitored and the response can be viewed in real-time, as shown in Figure 3c, simplifying the instantaneous analysis and debugging of a realistic network failure.

For our *real-time* interactive demonstration, we utilise iperf to generate UDP and TCP traffic, and we ask the conference attendees to exploit the preference and filtering capabilities of MiniNAM to adapt and alter the traffic flows. A detailed tutorial on how to build and run MiniNAM to replicate and repeat these examples is available at [2]. On the webpage, we offer additional scripts, with inadequate code, which illustrate how MiniNAM can be used to determine when the code is logically incorrect or insufficient to create the required network functionality. We provide a MiniNAM binary file on the webpage which is capable of all the functionalities mentioned in this paper.

Acknowledgement: This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 13/IA/1892. Special thanks to Darijo Raca and Alexander Revyakin for their advice and assistance in this work.

References

[1] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu and H. Yu. *Advances in Network Simulation*. Computer, Vol. 33 No.5, May 2000.

[2] B. Lantz, B. Heller and N. McKeown. *A network in a laptop: rapid prototyping for software-defined networks*. Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, October 2010.

[3] D. Estrin, M. Handley, J. Heidemann, S. McCanne, Y. Xu and H. Yu. *Network Visualization with Nam, the VINT Network Animator*. Computer, Vol. 33 No.11, November 2000.

[4] R. R. Fontes, A. L. Oliveira, P. N. Sampaio, T. R. Pinherio and R. A. Figueira. *Authoring of OpenFlow networks with visual network description (SDN version)*. Proc. Summer Simulation Multiconference, July 2014.

[5] J. J. Quinlan, A. Reviakin, A. Khalid, K. K. Ramakrishnan and C. J. Sreenan, *D-LiTE-ful: An evaluation platform for DASH QoE for SDN-enabled ISP offloading in LTE*. Proc. of the 10th ACM International Workshop on Wireless Network Testbeds, Experimental evaluation & CHaracterization (WINTECH), October 2016.

[1] https://github.com/OpenState-SDN/ryu/wiki/Server-Load-Balancing

[2] http://www.cs.ucc.ie/misl/research/current/ivid_demo/mininam/