A scalable architecture for emulating dynamic resource allocation in wireless networks

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>As Published</td>
<td><a href="http://dx.doi.org/10.1109/MILCOM.2009.5379801">http://dx.doi.org/10.1109/MILCOM.2009.5379801</a></td>
</tr>
<tr>
<td>Publisher</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>Version</td>
<td>Final published version</td>
</tr>
<tr>
<td>Accessed</td>
<td>Tue Mar 12 09:11:33 EDT 2019</td>
</tr>
<tr>
<td>Citable Link</td>
<td><a href="http://hdl.handle.net/1721.1/59997">http://hdl.handle.net/1721.1/59997</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>Article is made available in accordance with the publisher’s policy and may be subject to US copyright law. Please refer to the publisher’s site for terms of use.</td>
</tr>
<tr>
<td>Detailed Terms</td>
<td></td>
</tr>
</tbody>
</table>
A SCALABLE ARCHITECTURE FOR EMULATING DYNAMIC RESOURCE ALLOCATION IN WIRELESS NETWORKS

Devanshu Mehta, Jerold Jaeger, Arthur Faden, Kim Hebert, Navid Yazdani, Huan Yao*
Massachusetts Institute of Technology, Lincoln Laboratory
Lexington, MA

ABSTRACT

With the increasing size and complexity of emerging military communication networks, low footprint emulation methods are required to evaluate protocols, algorithms and user performance. In this paper we describe a new emulation architecture that is scalable, modular, and responds to real-time changes in topology and link characteristics. We use this scalable emulation testbed to evaluate system performance in a simple Satcom network topology, and enhance it to emulate Dynamic Resource Allocation using distributed software architecture.

I. INTRODUCTION

Providing bandwidth-intensive, real-time, and ever more complex services at the network edge is becoming increasingly important in DoD tactical networks. However, today’s typical design, development, and integration methods are not sufficient to support development of network infrastructures that can provide these new services. In addition to the more stringent network infrastructure requirements demanded by these new services, new satellite communication technologies such as dynamic resource algorithms and advanced mobile routing protocols further complicate network design.

In the face of these new requirements and technologies, new design and integration methods are needed. Current network simulation tools such as OPNET and ns-2 assist development under large user counts and traffic loads but lack real-time operation and effects of integration with network communication equipment. Alternatively, network testbeds using real wireless equipment can be very expensive and do not scale well in a laboratory test environment.

Network emulation, as described in this paper, provides a low cost, high fidelity technique for the evaluation of large-scale wireless networks. In addition, the architecture proposed here allows dynamic changes in network topology, configuration and resources. Emulation techniques such as ModelNet [5] that were designed specifically to address the challenges of emulating wireless networks have been used effectively in the networking community. The general approach proposed in this paper could be used to emulate a large variety of networks, built from the perspective of emulating wireless networks, using Ethernet-based COTS equipment and standard networking software as nodes. This allows us to keep costs down while exercising proven technology, but also introduces new challenges for emulating satellite and wireless topologies. We also discuss techniques for emulating wireless link and node characteristics such as loss, mobility, and channel interference.

We then augment this architecture to evaluate a specific technique—Dynamic Resource Allocation (DRA) in future satellite communication networks. A distributed control architecture is used to emulate (in software) the entities of the network to provide external stimuli to the network based on traffic demand, Quality of Service, weather, mobility and external algorithms for DRA.

We also demonstrate techniques through which users can trade scale for accuracy of the emulation. This includes the virtualization of interfaces and nodes and different link emulation techniques. For example, much higher scale can be achieved if only link rate is to be emulated as opposed to a more accurate emulation of latency, jitter, error rates and link rate.

II. EMULATION OF A WIRELESS NETWORK

The objective of this architecture is to use real network and routing hardware and software to build the network, while emulating links, link-layer characteristics, mobility, channel characteristics, resource allocation and wireless hardware such as modems. This allows us to test higher layer network functions such as routing, quality of service, TCP performance, and user applications over an emulated wireless network.

Standard networking equipment along with some external software is used to emulate the network. The link characteristics are emulated using egress rate shaping or stand-alone link emulators. Mobility is emulated as a

* This work is sponsored by United States Air Force under Air Force contract #FA8721-05-C-0002. Opinions, interpretations, recommendations and conclusions are those of the author and are not necessarily endorsed by the United States Government.
topology or configuration change in the emulated network.

In order to build our emulation architecture, the design was broken into three parts: the network platform, the control architecture, and control algorithms. A control algorithm, such as DRA, monitors the status of elements in the network and makes changes to the network. This is accomplished through a network of software agents, each agent representing a node in the network. The control algorithms and details of the control architecture are discussed in the next section, within the context of DRA and emulation of a satellite network.

The network platform is the collection of networked nodes, built using common network software and hardware components while emulating the links and topology. The network platform must have the ability to dynamically vary link characteristics, change the network topology based on mobility of the nodes, and the ability to emulate a large number of nodes without burdensome complexity in management or configuration. The platform should also support higher layer protocols such as routing and network capabilities such as quality of service. Finally, this platform must have the ability to provide statistics for each of the network elements and make changes dynamically, as often as several times a second, to emulate DRA.

One major advantage of abstracting the network platform from the control agents is that this allows the platform to support many different kinds of hardware and operating systems, as long as the requirements mentioned above are met. A variety of different types of network software can be used in the emulation with little added complexity.

The network platform consists of virtual or physical nodes, each of which represents a node in the network being emulated. A physical node is an actual router or device actively participating in the emulation. A single physical device can also emulate multiple virtual nodes through techniques such as virtual interfaces, virtual machines and containers. The correct balance of virtualization methods depends on the desired trade-off between scale, performance, and the type of network being emulated. Certain virtualization techniques provide higher accuracy and performance, with lower scaling potential. In a physical node, only the link is emulated, but the actual network node is real and is seamlessly connected to the emulation.

Figure 1 represents a simple emulated network where three physical nodes emulate a total of eight emulated “virtual nodes”. In this example, the emulation can be done using VLAN interfaces, virtual machines, software-based containers, or other software emulation techniques.

![Figure 1. Design of Emulation Network with Multiple Virtual Nodes Emulated on a Single Physical Node.](image)

Emulating a wireless network using standard Ethernet-based device is somewhat challenging, but is an important part of the architecture. Ethernet devices have the advantage of being cheap and easy to deploy in large scale with well-understood properties. In a satellite network, the point-to-point uplink and point-to-multipoint downlink introduces behavior that is counterintuitive in an Ethernet network. Mobility, link outages, link-layer and channel characteristics are features of wireless networks that require additional care when being emulated.

The simplest technique for emulating a large number of nodes on a single physical machine is to use VLAN interfaces on COTS routers along with private VLANs on an Ethernet switch and shaping on egress interfaces of each node. We use this approach to quickly set up an emulated network to test our control architecture. In the emulation of a satellite network, we use a VLAN as equivalent to a Downlink Multiplexing Group (DMG) for the downlink direction. This simplifies the problem of emulating a point-to-point uplink and point-to-multipoint downlink simultaneously using Ethernet-based devices.

When emulating a network using VLAN interfaces, it is possible that two interfaces on the same device are emulating nodes that are not directly connected. In order to force the device to forward packets out the correct interface and correctly emulate the desired topology, explicit source-based routing is used. In Figure 1, if VN1 and VN2 are emulated terminals on the same physical machine and VN5 is an emulated satellite router on a different machine, traffic from VN1 to VN2 must be forced to route through VN5. Software and routing techniques force VN1 and VN2 to behave as separate nodes.

The constraint on an architecture based on VLAN interfaces is that two adjacent nodes cannot be represented...
by virtual interfaces on the same node. Also, routing protocols and multicast are harder to test when using virtual interfaces for emulation. Using VLAN interfaces cause limitations in topologies, and routing and multicast testing. However, they provide a quick and low cost method to scale the emulation. Specific software techniques discussed in section III deal with some of these limitations effectively.

The preliminary emulation testbed was built based on COTS equipment. A Cisco 7301 software router [7] using IOS version 12.4(20)T was selected, based on the findings in [3], for the advantages it provided in cost, protocol support, and demonstrated techniques for supporting DRA. This router can support the polling, shaping and QoS required to emulate a wireless network effectively. The COTS architecture presented here has been used as a preliminary proof-of-concept in order to test different ways to emulate aspects of a wireless network, to exercise the DRA algorithm and the control scheme presented in the next section.

Dynamic configuration and monitoring of the network node is a required feature of our network emulation. With the Cisco router this can be achieved through telnet sessions and SNMP. In order to parallelize telnet commands to the node, especially when a single physical router emulates multiple virtual nodes, multiple sessions are utilized. This approach does not scale to large numbers, but 40 nodes per physical Cisco router has been achieved.

We found that SNMP on the Cisco 7301 for polling queue statistics was not well supported. The SNMP index for a particular queue is difficult to determine and may change on router reboot. Also, it is not currently possible to change egress shape rates using SNMP on this router, which makes the command-line interface a necessity for our emulation.

The COTS platform does not provide link emulation capability or hierarchical QoS queues, and so has been replaced with Linux routers in subsequent iterations of the emulation. The hierarchical queues are important when implementing DRA.

The Linux router approach scales extremely well (Section IV), when used in conjunction with Linux traffic control. Linux traffic control enables hierarchical queues and filtering the traffic in to multiple terminal-specific queues, removing the need for VLAN interfaces. Specific implementation of these approaches in the context of DRA is detailed in Section III. It is possible that virtualization of routers and software such as Linux containers can be used in addition to traffic control to provide better routing and multicast protocol support but this approach is yet to be investigated.

The advantage of the architecture proposed in this paper is that all of these approaches for emulating network nodes can be combined for varying degrees of scale. Virtual machines provide high accuracy in protocol emulation but the performance is not as good. VLAN interfaces are limited in protocol emulation support, but dramatically increase the ability to scale. A single emulated network with a set of Cisco routers with VLAN interfaces, a handful of Linux virtual machines, and a few actual physical routers with real interfaces can be combined, where the low fidelity virtualized nodes provide the scale and the high fidelity physical machines can be studied for performance.

### III. EMULATION OF DYNAMIC RESOURCE ALLOCATION IN SATCOM

Satellite communications systems look to provide high utilization of limited resources for a large number of terminals. The propagation delay, coupled with channel conditions that change frequently due to weather and mobility of nodes complicate the efficient utilization of resources. A centralized Dynamic Resource Allocation algorithm [1] is used by the system to share limited RF resources. DRA resides in each satellite and makes assignments to the satellite and terminals on an epoch-by-epoch basis. A terminal is assigned its uplink transmission mode and a set of timeslots in which the terminal can transmit. The transmission mode (a burst rate and modulation) when combined with the number of timeslots provides a rate that must be emulated on each node in our network.

Two components of DRA should be emulated in our system—Dynamic Coding and Modulation (DCM) and Demand Assigned Multiple Access (DAMA). DCM is the dynamic assignment of uplink and downlink resources to the satellite and terminal based on link conditions, demand in each of the traffic classes and their priority and general system resource availability. During good weather, a terminal may use bandwidth efficient modes, but can switch to lower rate power efficient modes during bad weather. DAMA is used on the uplink to provide fairness between terminals while supporting QoS.

In such a dynamically provisioned satellite, the link layer and network layer must work together to provide QoS, while each operates on differing timescales [4]. Also, in traditional QoS, the output rate of a router interface is assumed to be constant. In a system with DRA, the links vary with time, which has significant implications on QoS.
scheduling performance and the performance of end-user applications and TCP.

This presents many novel challenges for our emulation (Figure 2). First, we must calculate and apply the DRA assigned link rate to each emulated link in the network. This consists of the uplink interface of each terminal and the downlink interface of each satellite DMG. The rate of each link may change as often as once every epoch.

In addition to the link rate, we also emulate the DRA agents that reside in each of the terminals and the payload. The agents require queue statistics for each traffic class at each uplink and downlink interface. These statistics are provided to the DRA algorithm once every epoch to calculate the resources assigned, which translates to an emulated link rate. A modular and distributed architecture is required to emulate the control messaging between the agents (Figure 2) in order to successfully study the behavior related to control protocol in a dynamic environment.

Finally, the system must respond to external events such as weather, link outages and mobility of nodes. Weather causes signal-to-noise ratio changes, which directly influence the DRA algorithm. Mobility events can affect the network topology as well as DRA.

In order to emulate DRA in a satellite network, a real node or terminal is now emulated as two entities—a network node and a control agent. The network node acts on the data packets as a real router would. The control agent is a software emulation of the node, which passes control messages to other control agents and to external algorithms. The control agents run independently, in a distributed fashion, and do not need to be run on the same device as the network node. Each agent communicates with its network node to affect its rate allocation or other configuration.

In effect, the control agents serve as a translation layer between the inputs from external factors—such as DRA, weather, mobility, and other wireless-specific events—and the actual network nodes. In the reverse direction, the control agents allow the external algorithms and scripts to monitor the status of the network, queues, and links.

### A. Emulation Network Platform

In a study on integrating COTS routers into terminals with DRA, [3] suggests two methods for supporting dynamic changes in egress rate to match the RF channel, while providing QoS: pause frames alone and traffic shaping with pause frames. Since our emulation does not have a wireless link or hardware, the rate changes themselves will approximate the DRA assignments that the algorithm provides.

For our emulation study we chose egress traffic shaping alone as the technique for DRA link rate emulation for its simpler implementation due to the lack of additional equipment for generating Ethernet pause frames. This trade-off between fidelity of pause frames and scale provided by shaping is acceptable, since the emulation does not need to operate on the same time scales as the DRA timeslot assignments to a terminal modem. Also, mobility events and link outages can easily be emulated by reconfiguring the nodes dynamically.
collection of queue statistics every epoch. We discuss these performance results in Section IV.

A customizable, more accurate Linux-based network router emulation architecture has also been developed, which provides many advantages. This approach avoids using VLANs, and partitions the traffic for each virtual terminal using only Linux traffic control filters. A single Linux interface with a hierarchy of filters and queues can be configured to represent a large number of virtual terminals. The rate and delay emulation is achieved through a combination of traffic control and netem. All the physical nodes participating in the network are connected to a single switch, and PVLANs are used to partition the traffic on the switch. This simplifies the design and provides more control over QoS, polling and configuration of each virtual terminal.

The challenge of gathering statistics and shaping egress rates on a Linux-based node is similar to the previously described Cisco node. The added advantage of our Linux node is that more fine-grained software can be developed to report the statistics, without having to rely on the poor scaling capability of the command-line interface or the lack of support for SNMP.

The Linux-based approach described above can be enhanced with additional methods to achieve scale. It may be run in a virtual machine, an approach that loses performance but allows multiple router instances to be run on the same hardware. Another approach, that may increase processing efficiency and scalability, is to use Linux container technology such as OpenVZ to emulate multiple routers on the same operating system instance. These techniques require further investigation.

B. Emulation Control Platform

Using Linux based nodes provides us with the network architecture to emulate the satellite network, but a structure is required for the control and algorithm that forms the heart of DRA. For this purpose, each node in the network has a set of software agents associated with it (Figure 4). These agents operate independently of each other and have multiple responsibilities. For uplink DRA, each terminal software agent monitors the queue statistics on its associated terminal node and passes these statistics to the satellite agent every epoch. The satellite software agent packages all of the statistics received from the terminal agents and sends them to the DRA algorithm and receives assignments for each terminal in return. The satellite agent sends these assignments to their respective terminal agent. And finally, the terminal agent applies this assignment to the uplink of the terminal node. Each of these events happens as often as once per epoch.

Downlink DRA is similar, however processing is contained within the satellite software agents. A satellite software agent collects the queue statistics on the satellite node associated with it and sends it to the DRA downlink algorithm. The agent receives assignments for each DMG in return and applies them to the satellite node.

In addition to these agents, there is an administration agent that can provide external scripted events to the system such as weather changes and link outages. This agent also allows the configuration of a control message transmission delay so that statistics gathered from a terminal can be delayed before they reach the satellite agent and assignments delayed before being applied by the terminal agent. This delay is essential to emulating the effect of propagation delay on the DRA control algorithms. Also, all of these agents require a common notion of time to synchronize the epochs, which is provided through an epoch time server.

The distributed uplink control agent system is shown in Figure 4, where each emulated terminal is represented by an actual router, a router agent to poll and shape the router as well as a terminal agent which forms the interface between the router agent and external agents. Each emulated satellite is represented by a physical router, one terminal messaging agent for each terminal with which it communicates, the main satellite agent as well as the DRA algorithm. These components are grouped logically, to mirror the real satellite network, but may reside on separate real or virtual machines.

Although network nodes can be built using a variety of different platforms, such as Cisco IOS or Linux, the control architecture must support them seamlessly. This is handled through platform-specific router agents that talk directly with the devices, but provide a common interface to the main software control agent.
Each of the agents is run as a separate process and was built using Tcl/Expect. They pass messages among themselves using simple ASCII formatted messages using TCP, which enables each agent to be run on a separate machine and ensures that the connection state is known to each agent. If a link outage for a particular terminal is to be emulated, its associated software agent can easily stop communication with the rest of the agents and the system will gracefully continue. The interface of the agent with its associated network node depends upon the type of node. Telnet and SNMP on Cisco and Linux are provided as examples, but it can easily incorporate any network device and software interface through which queue statistics and link rate emulation can be supplied.

DMG changes involve an additional level of complexity due to the nature of Satcom links, where the uplink is point-to-point but the downlink is point-to-multipoint. An Ethernet switch is connected between the satellite and terminal nodes and one VLAN is configured per DMG. When a terminal changes DMGs, it may require a configuration change on the switch as well as route changes on the network node for the affected terminal as well as the satellite.

IV. PERFORMANCE RESULTS

There are two challenges to achieving performance of the DRA emulation environment—the timely occurrence of emulated events within an epoch and the accurate emulation of network elements under load. The number of terminals, the topology and the traffic load affect increase the complexity of these challenges. In this section, we discuss the performance of each of the elements of the emulation and the trade-offs in balancing scalability with fidelity of the emulation.

A. Emulation Network Platform Performance:

Certain events must occur in each epoch in an accurate emulation of dynamic resource allocation. For a large number of virtual nodes on a physical node, the ability to meet epoch constraints becomes a challenge. Each virtual node needs to report its statistics to the DRA algorithm and apply its assignment at most once per epoch.

In Figure 5, we show the performance of a Linux router in polling the statistics using traffic control. As the number of terminals increases, the amount of time taken to poll grows linearly. We use a simplistic approach of polling each virtual terminal serially, which is the worst case, but this does provide guidance in selecting the appropriate number of virtual terminals to emulate on a single physical node. Figure 5 also shows the performance of the same router in configuring the assignment using Linux traffic control. These tests were run on a Debian 4.0 workstation (Quad Core Xenon E5430 2.66GHz processor, 4GB of RAM). Improvements to Linux traffic control and the parallelization of requests have been implemented to increase the speed of configuration and monitoring. We were also able to run tests on the Cisco router with up to 40 emulated virtual terminals on a physical router at low data rates. Scaling the emulation depends on the length of the epoch and the time budget allowed for each event to occur. If the algorithm execution requires only a small fraction of the epoch, it leaves more time for requests and assignments to each of the terminals.

The aggregate data throughput expected for each virtual terminal will also help determine the number of terminals to emulate on a physical node. This will be constrained by the capacity of the Ethernet interface. The scalability considerations for the router emulating the satellite are not as challenging since it is not virtualized.

B. Emulation Control and Algorithm Performance:

The algorithms used in this study satisfy a reasonable time budget with a very large number of terminals. For the downlink algorithm, data for 1024 terminals can be processed under 30ms and for uplink 1024 terminals could be processed in under 60ms. The key in selecting and tuning the algorithm is to determine a time budget that fits within the epoch, leaving sufficient time for the remaining events. With a budget of 100 milliseconds for the DRA algorithm, the algorithm is able to process over a thousand terminals.

Since control is distributed, the major bottleneck is inter-agent and intra-agent communication, which is handled through Unix sockets within an agent and TCP messages between agents. In tests of terminal agents, we have successfully managed 1000 terminal agents from a single Linux workstation (Quad Core Xenon E5430 2.66GHz processor, 4GB of RAM) without significant performance degradation.
V. FUTURE WORK

The main purpose of the COTS implementation of the network emulation platform was to fully exercise the software agents and DRA algorithm. The restrictions on number of emulated nodes per router and limited link emulation capabilities have driven our development to more customizable Linux based network nodes. The initial Linux network platform described in section III is being built upon in order to support a large number of emulated nodes per operating system instance. This approach can be coupled with virtualization to provide scale and flexibility. The ability to control the software on the node and container-based virtualization improves the accuracy and efficiency of the emulation. Container-based virtualization in Linux (or network stack virtualization in FreeBSD) creates multiple isolated containers on a single physical server ensuring that applications do not conflict. This enables better emulation of complex topologies and protocols while keeping the cost of hardware low. We are also investigating the use of an open source data plane switch, which will allow better link emulation capabilities in the switch.

The link emulation capabilities in the COTS platform are very limited and can be dramatically improved using software in Linux. Another advantage of the emulation is that it can also be seamlessly integrated with a real network or network device under test. Other COTS routers with a wider feature set can also be employed in the emulation. Finally, large-scale tests with user applications and other tests similar to the OPNET simulations presented in [2] and [6] are under development to run on the emulation to observe the effect of various DRA algorithms on the performance of protocols such as TCP.

VI. CONCLUSION

Within this paper, we first briefly motivated the need for a wireless emulation testbed architecture. An implementation was then presented that dealt with the requirements of a wireless environment emulated with Ethernet-based devices. Specific modifications, including additional software based emulation, were presented that addressed unique requirements of future military Satcom networks including Dynamic Resource Allocation and satellite link characteristics. Initial benchmarks and tests were run to demonstrate the correctness and performance characteristics of the system within expected limitations. Options are being explored to improve the fidelity, scale and protocol support of the emulation architecture.

VII. REFERENCES


