A Calibration Methodology for Networked Cybersecurity Testbed Environments

Vineet Ghatge
USC

Prateek Jaipuria
USC

Alefiya Hussain
USC

Structured Abstract

Background: Researchers widely use networked cybersecurity testbed facilities, such as emulab-based testbeds, to conduct scientific experiments. Experimentation tools provide a high level logical view of the testbed resources that abstract away the heterogeneity of the underlying hardware and software. These abstractions are highly effective in enabling cybersecurity experimentation. However, currently there are no tools to characterize the collective properties of the assigned resources, and to understand their impact on the results.

Aim: In order to conduct a scientifically valid experiment, there is a need to understand the properties of the assigned testbed resources which form the "scientific instrument" for the experiment. In this paper, we propose a calibration methodology to allow experimenters to characterize, interpret, and adjust the experimental results by accounting for the impact of the underlying resources.

Method: We developed a calibration tool to characterize end-to-end unicast and multicast loss and delay measurements for networked cybersecurity testbeds. We explore the impact of packet size, burst size, hardware type, and operating systems on the measurements. We conducted 2162 experiments over a period of four months. We explored the impact of default and tuned system configurations.

Results: Our results indicate packet loss is a function of packet size. Small packets, with burst size more than buffer size experience large losses when using default system configuration. Allocating larger buffers at the sender reduces the packet loss significantly, but increases processing delay within the kernel. Thus, based on the characteristics of the system under test, the experimenter can tune and calibrate the instrument and results.

Conclusions. The underlying testbed facilities and resources that form the scientific instrument in a networked cybersecurity experiment can significantly impact the results of the experiment. It is important to understand the characteristics of the underlying resources and calibrate the results correctly to ensure valid experiments.

1 Motivation

Over the last decade, networked emulab-based testbeds have become the prevalent platform among researchers and developers to evaluate the performance, security, and resiliency of prototype distributed systems. Emulab-based testbeds provide an open shared testbed facility. The experimenters use the facilities to conduct a series of experiments to evaluate their systems. Each experiment requires creating a representative environment. The environment includes a topology to create the interconnections between the different components of the system, and traffic to create representative load conditions that will be experienced by the system.

Each experiment is allocated hardware and software resources from the shared testbed facility. Experimentation tools use a NS-2 based experiment abstraction for creating different topological structures and configuring the experiment nodes. It abstracts the details of the underlying resources and allows the researcher to focus on the interconnections between the resources using nodes and links to represent the interconnection graph. Currently there are more than a 100 Emulab-based testbeds across the world that use this experiment abstraction [3]. For example, DETERLab is one of the largest cybersecurity testbed with 566 general computers located at Marina del Rey and Berkeley.

This experiment abstraction has been very successful. It allows experimenters to design the scientific instrument that is required for their evaluations very rapidly, using a range of tools [2]. It also allows them to repeat their experiments at a later time and port them to different emulab-based testbeds with ease. However, while it is well known that each experiment instantiation may be allocated a different set of resources on the testbed, experimenters rarely account for the impact of these differences in their evaluations.

In this paper, we propose the notion of each experiment instantiation as a *creation* of a *scientific instrument* that requires calibration. In Section 5, we show that when operating at boundary conditions of high load or a large number of receivers, the results can vary widely. Without proper calibration of the experiment resources, i. e., the underlying scientific instrument, it is hard to interpret the results or account for the differences seen between experiments.

2 Background

The DETER testbed [1] is an emulab-based shared infrastructure designed for repeatable experimentation in the domain of computer security. While most of the experiment discussed in this paper are done on DETER, the results can be widely applied to most emulation based experimentation in the domain of networking and cyber security research.

The DETER testbed consists of several nodes, switches, and routers that can be configured to create a specified topology. The experiment nodes are connected to two networks at all times; an isolated experiment network which is used to conduct the experiment and a control network that can be used to send control signals and collect measurements during the experiment. The control and experiment network have multicast and unicast support to send and receive messages.

Each experiment is created from the logical view constructed by the experimenter. The inherent network and operating system configuration constraints and invariants are typically not exposed, understood, and rarely accounted for in the experiment results. A bare minimum experimental setup, with a single sender and a receiver is constructed, and parameters such as packet size and hardware type are studied to enhance this understanding. UDP based multicast probing is the primary mechanism to understand the system's uncertainty. UDP provides us with flexible implementation and multicast helps us characterize the system when scaled to higher number of nodes. Our goal is to help the experimenter community understand possible imperfections in experiments under the influence of testbed configuration.

3 Contributions

In this paper, we propose to view the resources allocated to each emulation experiment as a scientific instrument. We develop calibration tools that allow capturing the properties of the allocated hardware and software resources that form an experiment. We show that this process is necessary to establish a formal baseline of the testbed resources so that experimentation results can be correctly interpreted, compared, and understood.

Emulation-based testbeds are highly heterogeneous environments. They are constantly evolving due to hardware and software upgrades. Also, federation and virtualization capabilities are widely used in testbeds and hence there is a growing gap between the abstract logical view and the physical view of the testbed resources. This growing heterogeneity in the testbed facilities makes it even more important to measure and calibrate the capabilities of the allocated hardware and software resources for networked cybersecurity experiments. The calibration process provides a way to compare and understand, the differences among results, of various runs of the same experiment, which is hard to do in the absence of such a calibration process.

4 Methodology

Networking experiments that are emulated on testbeds, tend to use both one-to-one and many-to-many communication modes. While the two communication modes operate distinctively in different applications, the underlying testbed facilities shared by these experiments are the same. An uncalibrated experiment conducted in such testbed facilities often leads to inconsistent results. We use DETERLab, which is an Emulab-based networked cybersecurity experimentation facility, to do our experiments. Figure 1 shows a schematic representation of the DETER testbed. Mostly, such testbeds are controlled by a web based logical view, which enables experimenters to visualize and construct networking experiments. Experimenters communicate and control experiments via the control network indicated by dark lines in Figure 1. The experiment network is an isolated component shared only between the participating nodes indicated by light lines within the *scientific instrument* of Figure 1.

4.1 Calibration Tool

The custom command line calibration tool transmits UDP packets of varying sizes in bursts. The *burst size*, i.e., the number of packets sent out in a chunk, varies from 100 to 5000 packets. The sender sleeps for 100 seconds at the end of each burst. This is defined as the *inter-burst time*. The packets are tagged sequentially by the sender application. The tool monitors losses at three points in the communication path, indicated by A, B, and C in Figure 2. System measurements are gathered at A and C, whereas network measurements are taken at B.

The sender constructs UDP probing packets with a header and a payload. The header has a fixed size of 12 bytes, while payload size varies between 50 to 1200

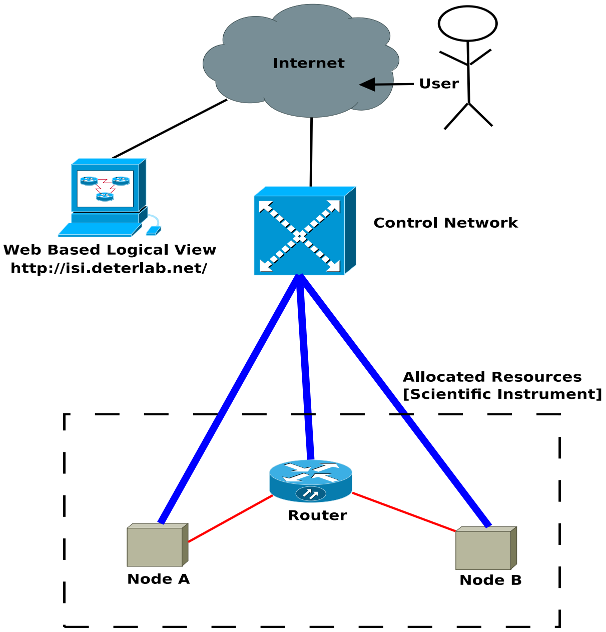

Figure 1: A schematic representation of DETER testbed.

bytes. The sender and the receiver applications communicate using UDP datagram sockets. The application code is implemented in C using Socket Programming API [5]. The sender sends packets based on the burst size and the packet size. Additionally, the tool activates tcpdump traces for verification and validation purposes.

Both the sender and the receiver application log, for each packet, the sequence number and send/receive timestamp (based on the gettime() API). Also, at each of the interfaces, we log the send/receive timestamp (using ioctl() function) for each packet. A snapshot of ifconfig and ethtool is captured before and after each packet burst, which helps us count the number of packets sent and received at the interfaces.

An asynchronous shell script gathers the sequence number logged at both, the sender and the receiver, to evaluate the packet loss, and plots histograms using a predefined gnuplot script. We calculate packet losses at three points in the communication path.

1. Packet loss between the sender application and the sender interface, indicated as A in Figure 2. This is calculated by comparing the number of packets sent by the sender application versus the number of packets logged at the sending interface.

2. Packet loss between the sender interface and the receiver interface, indicated by B in the figure, which is calculated by comparing the packets logged at the sender and the receiver interfaces.

3. Packet loss between the receiver interface and the receiver application, indicated by C in the figure. This is calculated by comparing the packets logged at the receiver interface against the number of packets received by the receiver application.

We additionally use the ethtool traces to verify the values obtained from ifconfig traces.

A python script gathers the timestamp information, from the logs at the sender and receiver applications, and the interfaces on both the nodes, to calculate delay. Each packet traverses the path from the sender application to the receiver application and back. The difference in time at which the packet is originally sent, and the time at which it is received back at the sender is used to calculate the total delay.

Using the features of DETERLab, we easily scale up the experiment to have more number of receivers. The experiment topology description file is modified to add the extra nodes. Along with that, the feature to run a command at node startup is used to run the receiver script on the additional receiver nodes.

4.2 Metric & Approach

To measure performance, we selected two control parameters and two operational metrics. The first control parameter is burst size or the number of packets sent in a chunk by sender at one time. The second is packet size or the total number of bytes that are wrapped in a single UDP packet. The operational metrics are loss and delay, which capture typical network performance behaviors. Loss and delay calculations, and the technique of measurements is similar to that adopted by Wang [6]. A preliminary multicast experiment was done for a group of nodes with a burst size of 5000 packets, each of size 62 bytes. Traces captured by the calibration tool indicate that packet loss is experienced between the applications and the corresponding interfaces. Further, there are no losses observed over the network between the sender and receiver interface. Based on our observation, we classify further sections into with and without tuning.

4.2.1 Without Tuning

Each experiment swapped into the testbed with a well defined NS-2 file is allocated a default configuration of software and hardware. A simple two node topology, described in the next section, is emulated on the testbed. The custom tool is then used to calibrate multicast and unicast experiments.

Multicast is a many-to-many communication protocol. One or more senders and a set of receivers, who subscribe to the same multicast group, should be able to send and receive packets. When the mentioned topology is emulated on the testbed, one node assumes the role of a sender, and the other node becomes the receiver. On the other hand, unicast is a one-to-one communication protocol, where the sender directly communicates with the receiver. With the help of the calibration tool, we explore the impact of packet size and burst size on operational metrics, and their perturbations are studied at three different points in the communication path.

4.2.2 With Tuning

An emulated experiment is an independent and fully operational system, on which the real world performance tuning measures are very well applicable. The participating nodes are tuned by allocating larger buffers, giving them higher operational capabilities, which otherwise do not exist.

We recalibrate the same experiment for multicast and unicast. A systematic comparison-based study is performed between the two communication modes for varying packet sizes and burst sizes. The imperfections demonstrated by the testbed are further explored by tuning tangible configuration details, like hardware type and operating system. At each step, we evaluate the impact on loss and delay, along with the system configuration conditions that dictate these behaviors.

4.3 Experimental Setup

As indicated earlier, the DETER testbed is made of a variety of nodes, switches, and links, that vary in many configuration aspects. The node types and convention used to represent them are listed in Table 1. Henceforth, the paper will use their representative names. Additionally, a detailed configuration specification is enlisted below.

1. 2.13 GHz quad core Intel Xeon processor based on a Dell PowerEdge 860 with 4 GB RAM, one quad port PCI gigabit ethernet card for the experimental network and one 250 GB SATA disk drive. These machines are available as bpc2133 and pc2133 on the DETER testbed.

2. 3 GHz dual core Intel Xeon processors based on a Dell PowerEdge 1850 with 2 GB RAM, 4 Intel Gigabit experimental network ports and one 36 GB SCSI drive. These machines are available as pc3000 and bpc3000 on the DETER testbed.

3. 3 GHz dual core Intel Xeon processors based on a Dell PowerEdge 1850 with 2 GB RAM, 5 Intel Gigabit experimental network ports and one 36 GB SCSI drive. These machines are available as pc3060 and bpc3060 on the DETER testbed.

4. 3 GHz dual core Intel Xeon processors based on a Dell PowerEdge 1850 with 2 GB RAM, 5 Intel Gigabit experimental network ports and one 36 GB SCSI drive. These machines are available as pc3060 and bpc3060 on the DETER testbed.

Hardware Type	Representative Name
pc3000	HW-A
pc2133	HW-B
bpc2133	HW-C
bpc2800	HW-D
bpc3000	HW-E
bpc3060	HW-F
pc3060	HW-G
pc2133x	HW-H
Microcloud	HW-I

Table 1: Representative mapping used for different hardware in Deterlab

5. 2.8 GHz dual core interl Xeon processor based on Sun Microsystems Sun Fire V60 Chassis with 2 GB RAM, 3 PCI-X Intel Gigabit experimental network ports and 36 GB SCSI Disk Drive. These machines are available as pc2800 on the DETER testbed.

6. 2.4 GHz quad core Intel Xeon E3-1260L processor based on a MicroCloud Chassis with 16 GB RAM, 5 PCIe Intel Gigabit experimental ports and 250 GB SATA Western Digital RE4 Disk Drive. These machines are available as MicroCloud on the DETER testbed.

During the course of the experiments, we work with three different experimental setups.

The basic case is a simple topology with two nodes, both of hardware type HW-A, logically connected by a 100 Mbps full duplex link with a drop tail queue, emulated on the DETERLab testbed. To avoid overhead delay and traffic shaping, delay nodes are excluded from the topology.

One of most extraordinary features of the testbeds like DETERLab is that an emulated experiment is easily portable across testbeds. Hence, for the next setup, we port the above-described experimental topology to another testbed named Emulab. On Emulab too, we use machines of type similar to HW-A. We also use the same parameters and metrics for calibration that we use on DETERLab. This also helps in demonstrating the extensibility of the calibration tool.

Finally, we do a set of experiments on DETERLab with a large scale topology consisting of one sender and twenty receivers, where the sender and the receivers are all connected through a simple 100Mbps LAN. The receivers consist of four different hardware types, HW-C, HW-D, HW-E, and HW-F, with five machines of each hardware type.

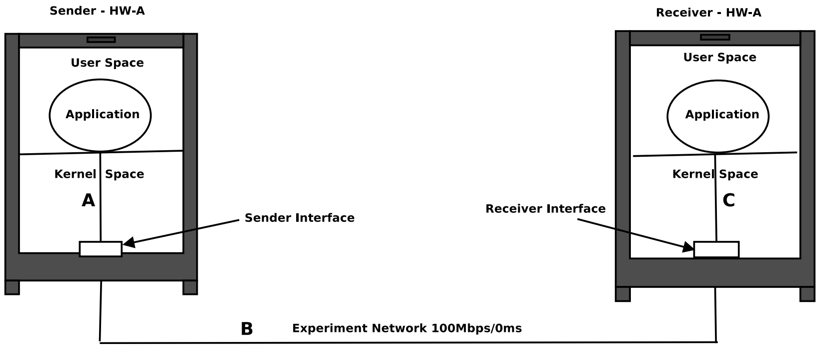

Figure 2: A schematic representation of experiment setup.

5 Results

In this section, we present our calibration results in two parts one for tuned and unturned systems. We leverage the operational metrics to understand network performance as defined in Section 4. We perform a total of 2162 experiments, out of which 1312 experiments are performed on DETER and remaining 850 experiments are carried out in emulab

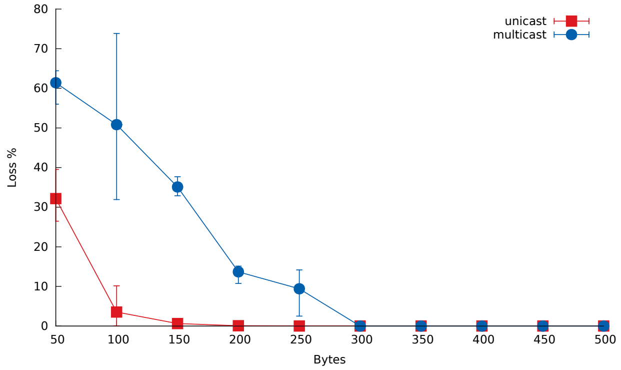

Figure 3: Loss Percentage between the Sender and Sender Interface

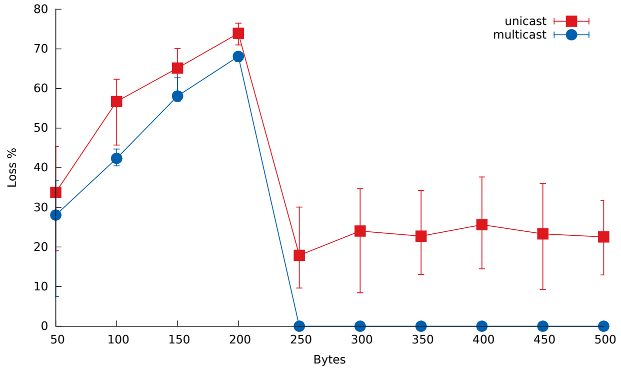

Figure 4: Loss Percentage between the Receiver and Receiver Interface

5.1 Without tuning

We conduct about hundred experiments with an untuned experimental setup described in Section 4. The calibration tool is used to perform a multicast and unicast experiments. The data collected are presented as error bar charts in Figure 3 and Figure 4.

The x-axis represents loss percentage or the number of packets loss between application and its corresponding interface. Similarly, y-axis represents control parameter - packet size in bytes. In Figure 3, multicast tends to experience more losses when compared to unicast for any packet size lesser than 500 bytes at the sender. However, the cause for this behavior is still under investigation. On the contrary at the receiver, multicast experience less losses than unicast as seen in in Figure 4. It is evident from both Figure 4 and Figure 3, that losses tend to taper down to zero beyond a packet size 250 bytes for multicast. Interestingly, unicast tends to be less aggressive than multicast, but there is a visible downside from 80% loss to about 18%. Packet size of 250 bytes is the knee point for a lossless network.

The reason behind this interesting behavior is that senders and receivers has 5 Intel Gigabyte experimental network port with an E1000 driver. Based on Intel specification, such a network port can handle about 500,000 packets per second. Its well known that rate of sending packets per second is determined by the number of bytes per packet. Based on this fact we construct a rate to bytes mapping. Table 2 which enables us or experimenters to understand rate limitations experienced by hardware network port. A knee point of 250 bytes and above, achieve a packet rate ranging between 536,870 - 268,435 which is well with the acceptable processing range of the Intel network port.

In order to present a wider perspective to experimenters, an additional 750 experiments are conducted across experiment and control network. Measured data is plotted in a 3D representation as seen by Figure 5 and

Bytes/sec	Packets/sec
50	2,684,354.56
100	1,342,177.28
150	894,784.85
200	671,088.64
250	536,870.91
300	447,392.43
350	383,479.22
400	335,544.32
450	298,261.62
500	268,435.46

Table 2: Packets per second for various packet sizes

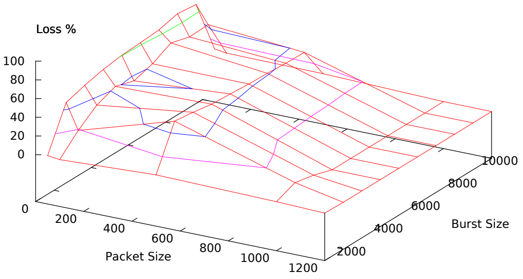

Figure 5: Application to Application Loss% in control network for varying packet and burst size

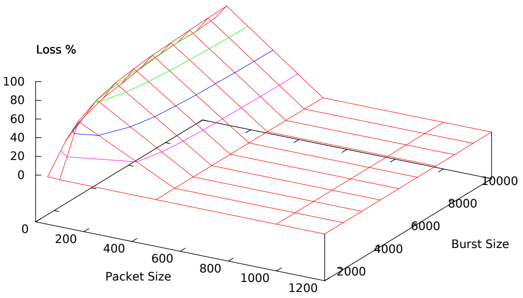

Figure 6: Application to Application Loss% in experiment network for varying packet and burst size

Figure 6. The x-axis and y-axis represent control parameters - packet size and burst size respectively. On other hand, the z-axis represents loss percentage. Each of the graphs provide users appropriate peak points of losses experienced by experiments for a combination of packet size and burst size. An experiment conducted across the control network experiences more losses than experiment network owing to the fact that the control is shared network across nodes and applications unlike the experiment which is dedicated to the physical emulated experiment of the testbed.

The set of experiments conducted so far elucidate that small packet size in an experiment i.e., lesser than 250 bytes experience higher losses than larger packet sizes. Additionally, control network has additional interferences which affect the result measured in experiments. In the next section, we tune the system to reduce this loss.

5.2 With Tuning

Linux distributions have default values predefined for socket descriptors buffer size and interface queue length. These values impact the behavior of the applications which are running on testbed like DETER. The tuning procedure involves allocation of larger transmission queue on sender like 10,000 using the ethtool. At the receiver, we increase the incoming reception connections to 65,536 and socket buffer space to 12,582,912 using the sysctl config file. Post the changes, about hundred experiment each is conducted for multicast and unicast calibration.

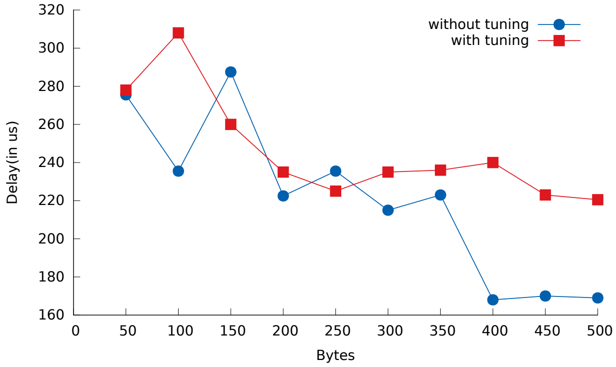

Figure 7: Delay variation for Multicast for with and without tuning scenarios

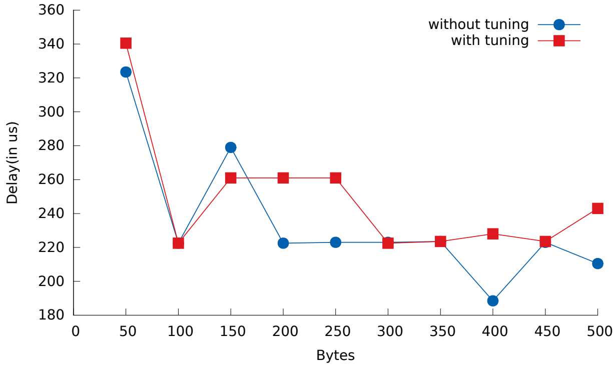

Figure 8: Delay variation for Unicast for with and without tuning scenarios

A quick analysis of data gathered by the calibration tool show that losses have dropped to zero. The y-axis on the graph is represented by delay in microseconds, however the x-axis on graph represents the packet size in bytes. Figure 7 show that delay comparison between tuned and untuned multicast experiments. Figure 8 shows the delay comparison for unicast experiments for tuned and unturned systems. From both graphs we can infer that tuned experiments suffer from longer delays than untuned experiments. Additionally, smaller packets experience longer processing delays than bigger packets. Smaller packets results in more number of packets received and sent at the interface. As result, the number of interrupt calls made by interface are higher. This adds an additional overhead on time spent in the end to end delay calibration.

5.2.1 Hardware Type

52 experiments are conducted to understand the impact of node hardware on experimental results. We conduct a multicast experiment on tuned and untuned systems with range of haradware type listed in Table 1. Figure 9 shows similar phenomenon as discussed in previous sections. Table 3 and Table 4 enlist the overall loss for tuned and delay calibration for untuned. Most of hardware experience losses for small packets and similarly, fixing loss on a experiment introduces delay which is high for small packets.

5.2.2 Operating System

To investigate the effects of kernel on experiments, keeping our experimental setup the same, we load FreeBSD 9.X operating system on nodes instead of default Ubuntu 12.04. The x-axis in graphs is the packet size in bytes whereas the y-axis is loss in percentage. Figure 10 display the loss seen at sender and sender interface for a multicast experiment for tuned and untuned scenario. Figure 11 displays the loss at receiver and its interface for the same experiment. FreeBSD has same knee point of 250 bytes like Ubuntu 12.04 as the hardware remains unchanged.

5.2.3 Interrupt

Interrupt is a signal to processor that a particular task needs immediate attention. We estimate the number of interrupts made to understand its implications on delay. We monitor interrupts from data collected in about 50 experiments and present them in Figure 12. The x-axis represents the number of interrupts whereas the y-axis represents byte size. The graphs shows that number of interrupts over a period of 5 microseconds for different packet sizes. The number of interrupts are higher for 250 bytes owing to more number of packets being processed at the interface. These interrupts contribute to overall delay experienced by the experiment.

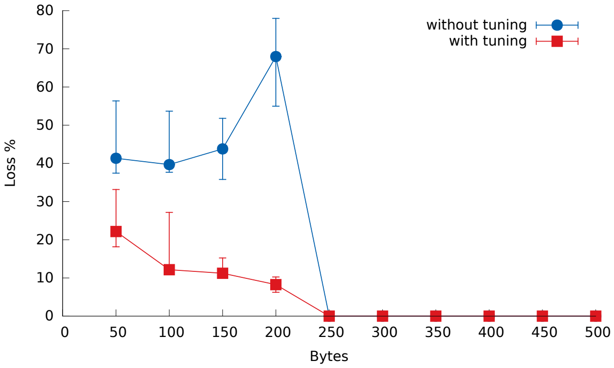

Figure 10: Loss between sender and sender interface for FreeBSD

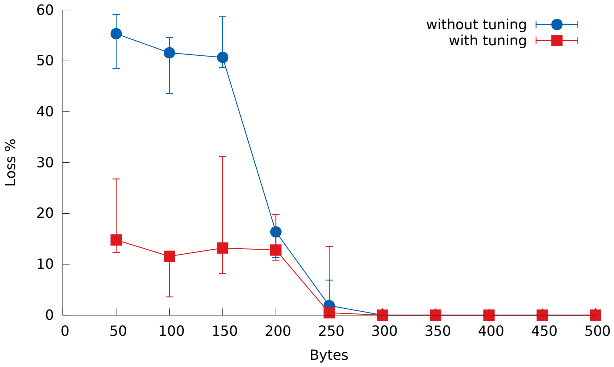

Figure 11: Loss between receiver and receiver interface for FreeBSD

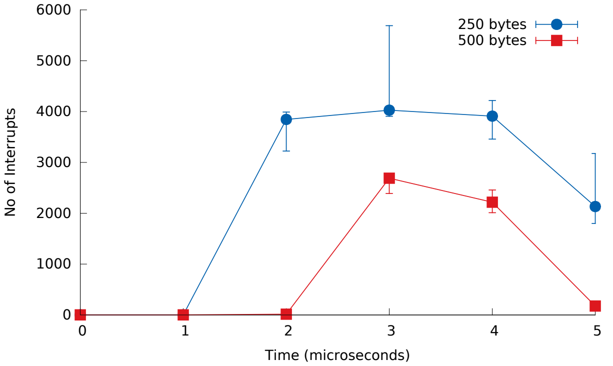

Figure 12: Number of Interrupts for different byte size in a multicast experiment

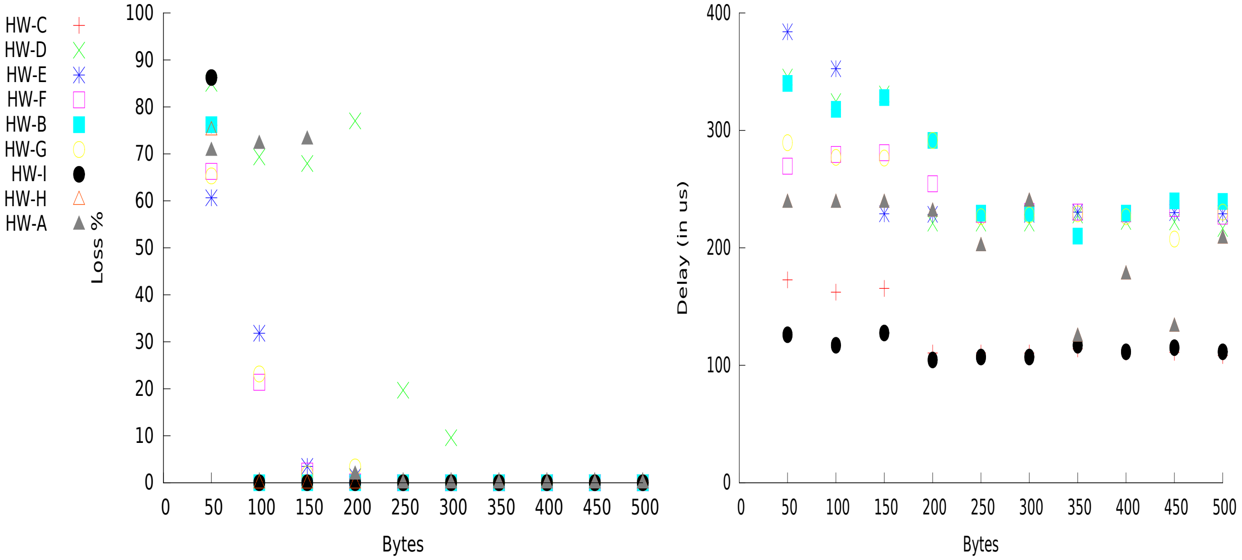

Figure 9: Loss and delay characteristics for different hardware type with and without tuning

Packet Size	bpc2133	bpc2800	bpc3000	bpc3060	pc2133	pc3060	Microcloud	pc2133x	pc3000
500	0	0	0	0	0	0	0	0	0
450	0	0	0	0	0	0	0	0	0
400	0	0	0	0	0	0	0	0	0
350	0	0	0	0	0	0	0	0	0
300	0	9.56	0	0	0	0	0	0	0
250	0	19.68	0	0	0	0	0	0	0
200	0	77.02	1.34	0.16	0	3.34	0	0	1.88
150	0	67.96	3.46	2.46	0	1.86	0	0	73.22
100	0	69.34	31.82	21.4	0	23.16	0	0	72.28
50	76.1	85.04	60.64	66.28	76.22	65.3	86.24	75.16	70.8

Table 3: Loss measurements for packet size between 500 - 50 without tuning

Packet Size	bpc2133	bpc2800	bpc3000	bpc3060	pc2133	pc3060	Microcloud	pc2133x	pc3000
500	108.25	216.5	229	227	239.5	230.5	111.5	208.5	208.5
450	111	222	230	233.5	240	207.5	115	133.5	133.5
400	111.25	222.5	229	229	229.5	226.5	111.5	178	178
350	114	228	230.5	230.5	210	229	117	125	125
300	110.5	221	228.5	231	229.5	228	107	240	240
250	110.5	221	229	228.5	229.5	226.5	107	202	202
200	110.5	221	228.5	254.5	291.5	291.5	104.5	231.5	231.5
150	165.5	331	229	281	328	276.5	127.5	239	239
100	162.25	324.5	352.5	279.5	318	277	117	239	239
50	172.75	345.5	384	269.5	340	289.5	126	239	239

Table 4: Delay measurements for packet size between 500 - 50 with tuning

5.3 Emulab

Testbeds experiments are expected to be portable across different testbeds. It is expected that these experiments yields same results across testbeds. To understand the portability and repeatability of experiments, a similar experimental setup as that on DETER is executes on emulab. Using the calibration tool, we execute multicast experiment for tuned and nontuned scenarios. The hardware type used in Emulab similar hardware type as HW-A. Table 5 gives a comparsion between HW-A type in Emulab and DETERLab

Configuration	Emulab HW-A	DETERLab HW-A
MachineType	Dell PowerEdge 1850	Dell Poweredge 2850
CPU	3.0 GHz Xeon	Dual 3Ghz Xeon
RAM	2 GB	2 GB

Table 5: Configuration of pc3000 in Emulab and DETERLab

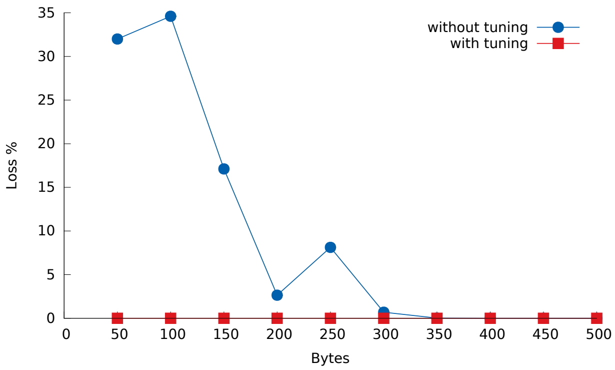

Figure 13: Loss variation for Multicast for with and without tuning scenarios

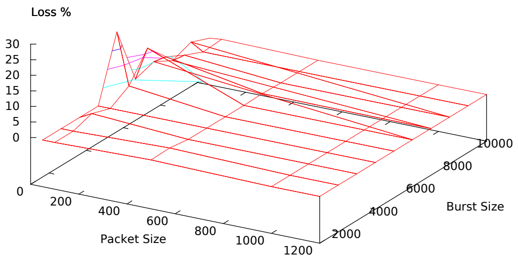

Figure 14: Application to Application Loss in Emulab Experiment Network for multicast applications

The x-axis in the graphs shows the loss percentage and y-axis shows the increasing packet size. Figure 13 shows that multicast behaviour is similar to that seen in previous results. The tuning procedure is valid and applicable to this testbed and no more loss is seen in the experiment as shown in Figure 13.

Similar to DETER, an 750 experiment is conducted acrosss the experiment network to explore calibrate the testbed. The graph use the same metrics as DETER 3D plots. Figure 14 shows a 3D plot of the loss behaviour for Emulab HW-A for varying burst size and packet size. The emulab experiment helps us understand that calibration mechanism introduced in the paper can be universally applied for all testbeds. The current state of art is to orchestrate federated experiment across testbeds. [4] However, given the possible imperfections for an experiment in a single testbed, an experimenter needs able to reconstruct his *scientific instruments* such that accurate results can be obtained.

5.4 Scaled Experiment

In this section, we move away from simple experimental setup described in Section 4 which was used to rationalize system behaviors. A large scale multicast experiment is orchestrated for 21 nodes - single sender and twenty receivers. Once the experiment has been emulated, the calibration tool is executed for rate of burst size starting at 200 till 1200 packets.

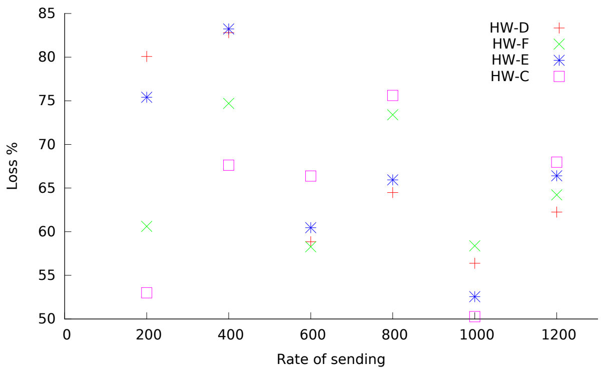

Figure 15: Loss % for multiple receivers in an experiment on DETERLab

The x-axis in Figure 15 is the loss percentage and y-axis is the packet size in bytes. The data points are average values per hadware type. The graph indicates that losses are non-uniform with rate of sending packets. An calibrated experiment such as this can result in uncertain estimation of values leading to incorrect intrepretation. The set of results so far present a wide perspective with a well-qualified reasoning about the implications of a imperfect testbeds and encourages experimenters to pursue calibration methodologies before executing any experiments.

6 Conclusion

In this paper, we presented a calibration methodology to enable the experimenter analyze and interpret the impact of the allocated resources on the experiment. Experiments are usually designed with graph abstraction that shield the experimenter from the increasing heterogeneity due to upgrades, federation and virtualization technologies in emulab-based tested. For scientifically valid experiments, it is important to correctly calibrate the allocation resources that form the scientific instrument for the experiment.

We calibrated 2162 multicast and unicast experiments on DETERLab and Emulab over a period of four months. We evaluated the impact of allocated resources on two operational parameters, loss and delay. We explored varying control parameters such as burst size, packet size, and number of nodes in the experiment. Our results indicate packet loss is function of packet size. Further, allocating large buffers reduce losses significantly, but impacts processing delay within kernel. Additionally, we study experiment portability across testbeds and how large scale experiments can be inconsistent in the absence of calibration. These observations indicate it is crucial to understand the characteristics of underlying resources in order to accurately calibrate results in an experiment.

References

[1] BENZEL, T. The science of cyber security experimentation: the deter project. In *Proceedings of the 27th Annual Computer Security Applications Conference* (New York, NY, USA, 2011), ACSAC '11, ACM, pp. 137–148.

[2] EMULAB. Emulab client interface. http://www.emulab.net/netlab/client.php3.

[3] EMULAB. Other emulab testbeds. https://wiki.emulab.net/Emulab/wiki/OtherEmulabs.

[4] FABER, T., AND WROCLAWSKI, J. A federated experiment environment for emulab-based testbeds. *Testbeds and Research Infrastructures for the Development of Network Communities 0* (2009), 1–10.

[5] STEVENS, W. R., ET AL. Chapter on multicast. In *UNIX Network Programming, Volume 1, Third Edition Source Code* (2004), Addison-Wesley Professional; 3 edition, pp. Pg 487 – 572.

[6] WANG, F., MAO, Z. M., WANG, J., GAO, L., AND BUSH, R. A measurement study on the impact of routing events on end-to-end internet path performance. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (New York, NY, USA, 2006), SIGCOMM '06, ACM, pp. 375–386.