Reliable Multicast in the DETER network

Vineet Ghatge
Anuj Gupta

April 12, 2014

1   Introduction

The DETER testbed [9] is shared infrastructure designed for medium-scale repeatable experiments in the domain of computer security. The testbed provides unique resources and a focus of activity for an open community of academic, industry, and government researchers working toward better defenses against malicious attacks on the world's networking infrastructure.

The DETER testbed [9] is made up 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 network has multicast support to send and receive messages from several nodes at once.

As Multicast [4] is UDP (User Datagram Protocol) based, it does not guarantee the delivery of a message stream. So the challenges faced by multicast are that the messages may be dropped, delivered multiple times , or delivered out of order. Our preliminary multicast experiments indicate that multicast network in DETER is unreliable. We aim at adding reliability mechanism to multicasting on DETER network to analyze its effect on the number of messages lost, size of the loss bursts and frequency of losses.

The current state of the art in DETER does not have any flexible and easy to use tools to measure the multicast loss bursts [1]. There have been several attempts in the past to emulate multicast in DETER, to the best of our knowledge we do not know if any of them have been successful so far. Our long term goals are

1. Understand and evaluate the performance of multicast network on DETER for varying combinations.

2. Implementing reliability in the network and attempting to study the impact of size, frequency and factors affecting loss burst for multicast networks.

This report presents an inital understanding of the DETER performance with respect to multicasting [7] [8].

2   Methodology

Multicast is a network addressing method for the delivery of information to a group of destinations simultaneously using the most efficient strategy to deliver the messages over each link of the network only once, creating copies only when the links to the multiple destinations split (typically at network switches and routers). A reliable multicast protocol adds the ability for receivers to detect lost and/or out-of-order messages and take corrective action (similar in principle to TCP), resulting in a gap-free, in-order message stream [10].

We will attempt to analyze experiment and control networks in DETER for the number of messages lost, size of the loss bursts and frequency of losses. Upon observing the unreliable behavior of both the networks, we will now add a simplistic reliability mechanism to multicasting in DETER lab. We will firstly attempt to add this mechanism for a one to one multicast network and then scale it up for a one to many multicast network. We will attempt to analyze its impact on performance, the size of the loss bursts and delay of the multicast messages.

The experiment is setup such that we have single sender and single receiver on the experimental network. The nodes have two NIC cards each of which are connected to the experiment and control network. We do not enforce any traffic shapping factors or parameters on the network, this ensures that no other parameters is going to affect the traffic flow across the link. We consider the following paramters to be crucial in affecting the network

1. Burst Size: This is defined as number of packets which are sent by the sender.

2. Packet Size: This is defined as the number of bytes per packet that are transmitted by the sender.

At each iteration of the burst size we evaluate loss burst, in order to facilitate and identify this: we construct a simple multicast packet which has a header of 12 bytes in size without the payload. One of the "fairly" missing factors

of UDP is the absence of sequence numbers. The reason we call this fairly is contrasting behaviours of TCP versus UDP, one would notice this one of the features which is characteristic of a connection oriented protocol [10]. We collect extensives amount of measurements such as - Sequence Number both at the sender and receiver, Nestat and Ethtools output. The latter two are required to monitor queues for any packet drops. Our Experiment is evaluated against both the control and experiment network. One would ideally expect the experiment network to perform better than control network. One of the main reasons is that control network has a fixed 100Mps link who's characteristics can never be controlled.

The multicast sender and receiver applications were written in C using the concept of Socket Programming API [10].To gather accurate data, we need to perform significant number of executions of our experiment and hence we wrote automation scripts. We wrote the following types of scripts

1. Shell scripts : To evaluate missing sequence numbers based on the intersection of sender and reciver trace files [2, 6].

2. Gnuplot scripts : Takes the sequence number of lost packets to plot the histograms [3].

3. MAGI scripts : To enable workflow deterministic control when we scale the number of receivers and senders [5].

Since UDP has no deterministic information to inform the receivers about the number of packets being sent we introduce the concept of round number. We define one round from first packet to last packet sent from the sender for a fixed burst size. The round number is field defined as part of packet header. We perform a linear increment in burst size for a fixed size in every execution of the experiment. The sequence numbers generated are serially increasing till the end of each round. An end of round is indicated by a conflicting packet round number as opposed to a expected packet number at receivers end.

3   Results

The experiment is conducted in two scenarios with the custom UDP packets constructed exclusively for experiments [1].

1. Experiment Network : We deployed the application to create a socket on one of the internal IP address - 10.1.1.X. We exchanged traffic across the link and evaluated the loss burst.

2. Control Network : We deployed the application to create a aocket on one of the external IP address - 192.168.X.X. The path taken by this mode need not be an immediate hop. Based on the different combinations of packet size and burst size, we generate ten simulations. The average of these simulations gives us good overview and consistency of the network behaviours. We study each of these keeping one parameter constant at time.

3.1 Packet Size

In this section we keep the packet size constant(in our case its about 72 bytes), and vary the burst size of 100 and incrementally build it by factor 100 to understand its small scale jumps. We evaluate the network behaviour till about 5000 packets on both experiment and control mode. Fig 1 shows behaviour of packet loss with varying burst size. We see that loss burst increments in a linear fashion. This behaviour is as expected. Loss rate is indicated by the averages accounted at every loss burst size(green line in the figures). Fig 3 is a plot in the similar lines as Fig 1, with burst size ranging between 1500-5000. The range between 4000 - 5000 in Fig 4 need some explanation as the loss reaches a constant change. This might be because of the devices being optmized for certain range values. We suspect this behaviour might continue and is an area to explore and understand.
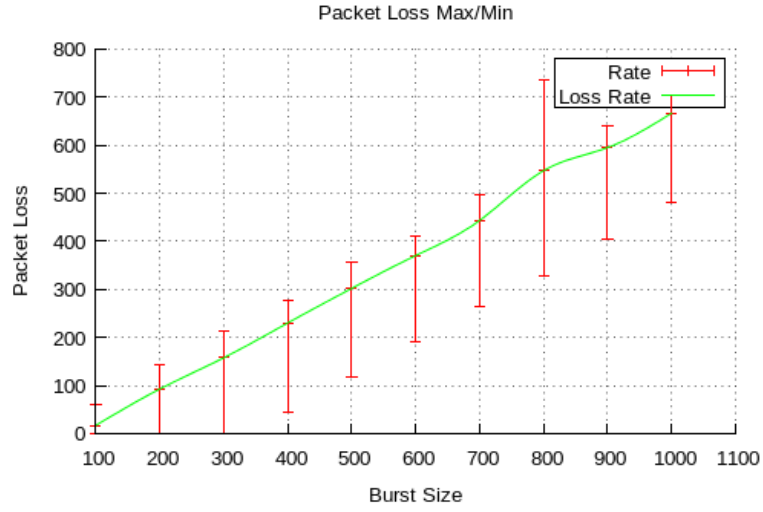
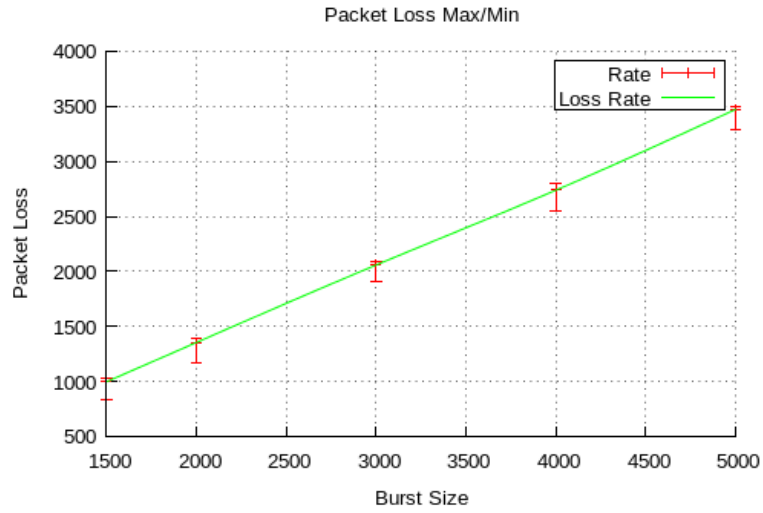Figure 1: Packet Loss on Control Network for Burst Size varying between 100-1000

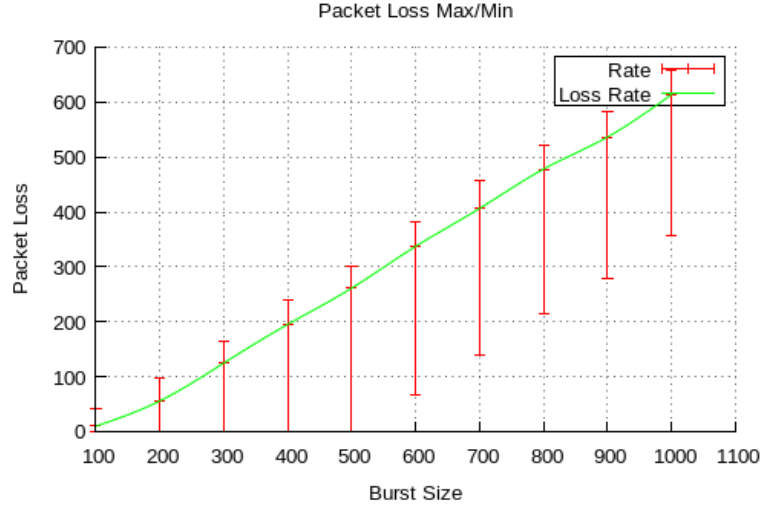Figure 2: Packet Loss on Control Network for Burst Size varying between 1500-5000

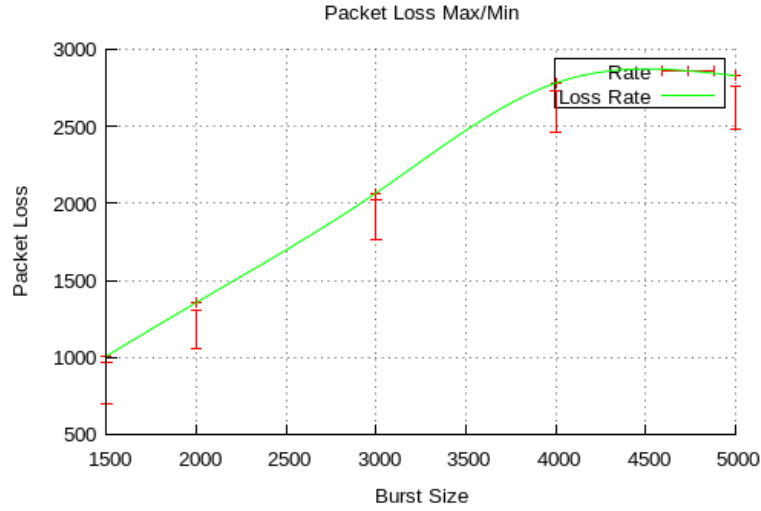Figure 3: Packet Loss on Experiment Network for Burst Size varying between 100-1500

Figure 4: Packet Loss on Experiment Network for Burst Size varying between 1500-5000

Contrasting the control network versus experiment network, it is evident that control network is far more decremental in loss burst. This is indicated

by the range of error bars which progressively shrink with packet size.

3.2   Burst Size

In this section we keep the burst size constant(same as above 5000 packets) and vary the packet size from 72 bytes upto 1212 bytes. We evaluate the network behaviour for both control and network mode. Notice, the comparison between Fig 5 - Fig 9, we do not see any loss burst till we reach above 100 packets. On average the loss burst varies from 40 to 100 packets for packet size between 72 bytes to 512 bytes.The losses are fairly consistent with progress of traffic in the control network.
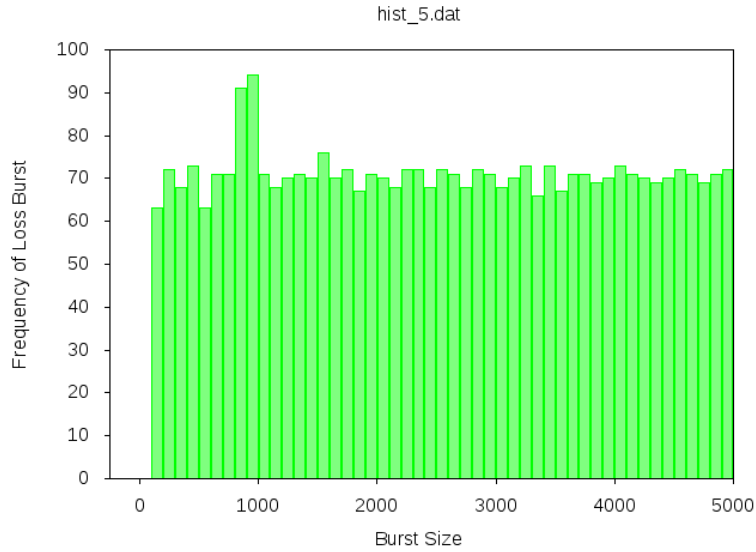

Figure 5: Frequency of Loss Burst on Control Network for 72 bytes and Burst Size of 5000 packets
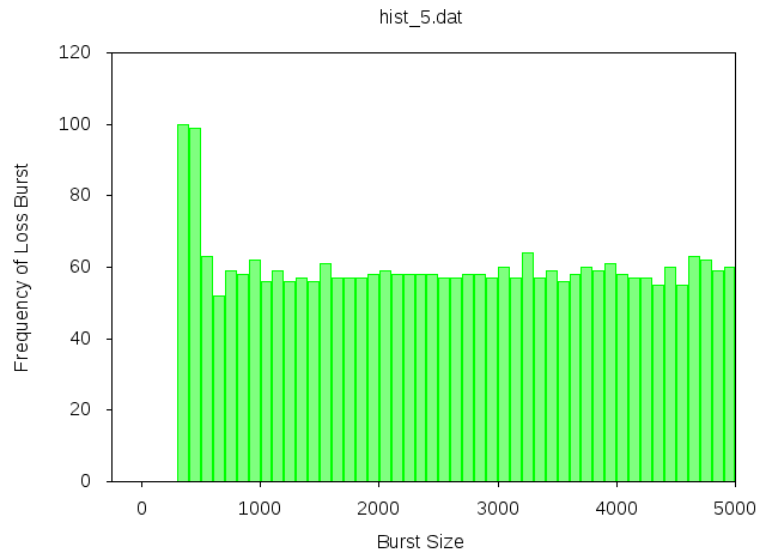
Figure 6: Frequency of Loss Burst on Control Network for 112 bytes and Burst Size of 5000 packets
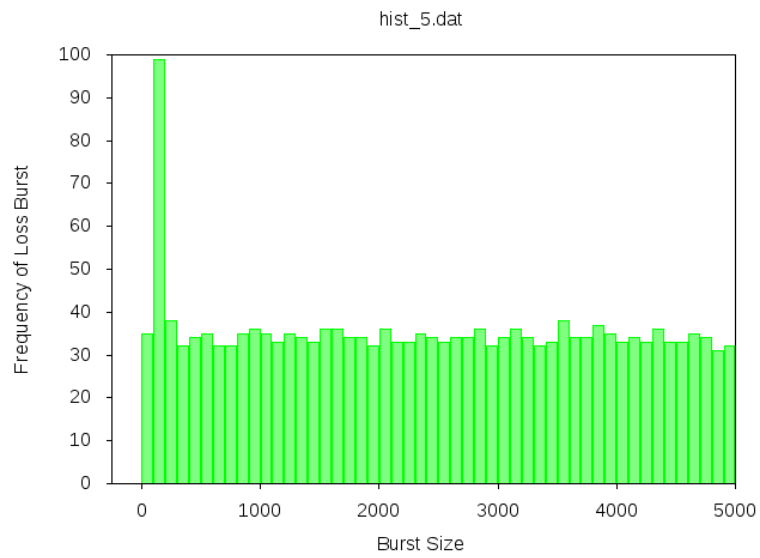

Figure 7: Frequency of Loss Burst on Control Network for 512 bytes and Burst Size of 5000 packets

In theory, extrapolating our results we expect the loss burst to worsen or stay consistent with packet size increase. However, our results from Fig 8 - Fig 9 show that loss burst diminishes. One of the reasons why we might be seeing this behaviour is due to the optimization of the switches on the network. Our results from experiment network show the similar behaviour, however the loss percentage are far lesser than control network.
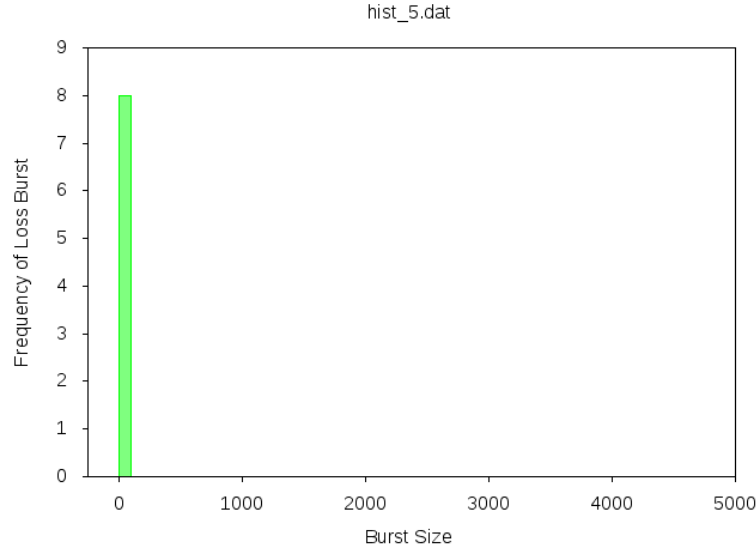

Figure 8: Frequency of Loss Burst on Control Network for 1012 bytes and Burst Size of 5000 packets
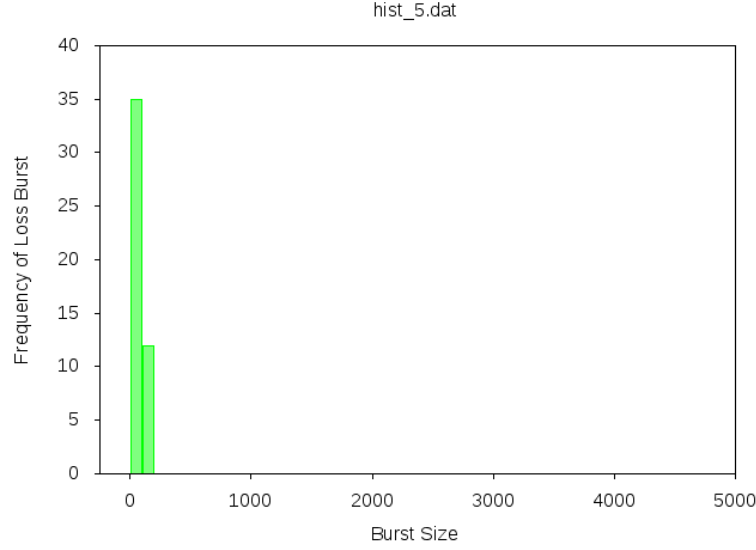
Figure 9: Frequency of Loss Burst on Control Network for 1212 bytes and Burst Size of 5000 packets

Our last graphs - Fig 10 and Fig 11 are 3D plot of data from all the experiments conducted. Notice the peak points in both the graphs which are indicative of "sweet" spots - point where the network has high degree of loss and this surge flattens out. These graphs have provided us useful data points in understanding how DETER behaves with multicast. Contrasting the two graphs - the control network degrades exponentially unlike the experiment network. Based on our analysis on DETER, we extended our experiment on Emulab. Our inital results suggest degraded performance, while we do not present the analysis here. The comparison might prove an useful baseline in understanding performance on DETER.
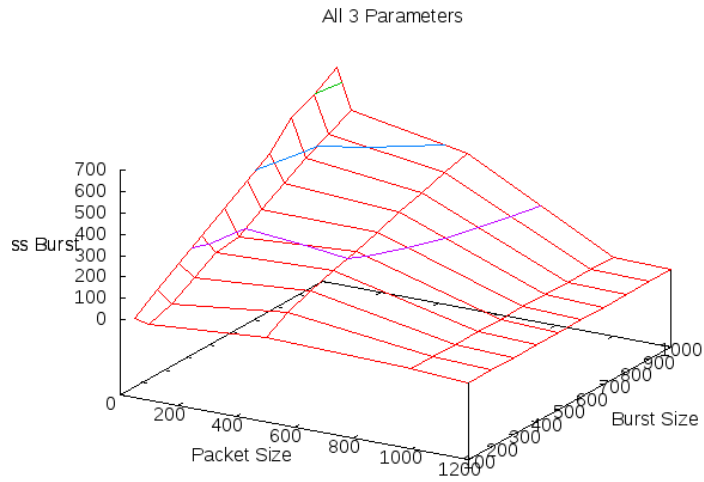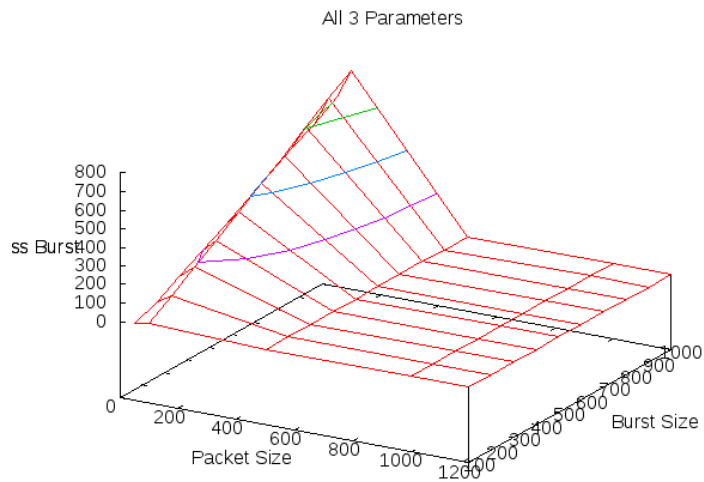
Figure 10: 3D variation for Control Network

Figure 11: 3D variation for Experiment Network

4 Summary and Path Forward

In conjunction to the experiments and results, we can conclude that the control network is less reliable than experiment network in DETER testbed. This follows logically, as we identify control network as a shared network and experiment network as dedicated network. So, we need to add a reliability mechanism for the control network. We also see an odd behavior of the experiment and control networks becoming more reliable as the size of the packet increases. Our initial speculation is that this behavior is due to an optimization mechanism in the routers and switches in the DETER testbed, which provide better network performance and reliability as the size of the messages increases. We need to analyze this behavior with more granularity to give a concrete and cogent explanation. Our preliminary experiments for Emulab to corroborate and compare results and findings for DETER testbed.

While our current results give us an conclusive understanding of the DETER testbed performance, our main goal of including relability in multicast and understanding its performance is a long path ahead. We identify our next challenges to be

1. Understanding the node types used in our experiment and showcasing machine packet loss versus network packet loss.

2. Understanding network behaviour for packet size between 112 bytes and 512 bytes.

3. Orchestrating a workflow for one to many multicast.

With extensive qualitative experimentation, we will have an better understanding of the degree of reliability that we will need to incorporate into the DETER testbed.

References

[1] Creating experiments on deter. `https://trac.deterlab.net/wiki/Tutorial/CreatingExperiments`.

[2] Ethtool - standard linux utility for controlling network drivers and hardware. `https://www.kernel.org/pub/software/network/ethtool/`.

[3] Gnuplot - command-line driven graphing utility. `http://www.gnuplot.info/`.

[4] Introduction to multicast. `http://www.mathcs.emory.edu/~cheung/Courses/558/Syllabus/14-Multicast/intro.html`.

[5] Magi - montage agent infrastructure.

[6] Netstat - command line tcp/ip networking utility.

[7] Pragmatic general multicast. `http://en.wikipedia.org/wiki/Pragmatic_General_Multicast`.

[8] Scalable reliable multicast. `http://en.wikipedia.org/wiki/Scalable_Reliable_Multicast`.

[9] D. M. Peter A. H. Peterson. Student introduction to deter. `https://education.deterlab.net/DETERintro/DETERintro.html`.

[10] A. M. R. W. Richard Stevens, Bill Fenner. Chapter on multicast. In *UNIX Network Programming, Volume 1, Third Edition Source Code*, pages Pg 487 – 572. Addison-Wesley Professional; 3 edition, 2004.