# A Distributed Ethernet Traffic Shaping System

Bannazadeh H., Leon-Garcia A.

Electrical and Computer Engineering Department
University of Toronto, Toronto, ON, Canada
Email: hadi.bannazadeh@utoronto.ca

## Abstract

*We present a Distributed Ethernet Traffic Shaping (DETS) system for regulating the flow of data when multiple virtual machines run on one host and share a single Ethernet link to send and receive traffic in a cluster or a data center. In such settings, virtual machines can undermine each other's ability to receive traffic on the shared link. In DETS, sending machines monitor and regulate their transmission rates to conform to target rates. The DETS system is implemented in the host system, and there is no need to change or modify any processing or networking hardware. We describe an implementation on a Linux-based cluster, and through experimental performance evaluations, we show that DETS can guarantee the access rate of virtual machines. We also describe modifications to the Ethernet control plane so that DETS can be natively supported in Ethernet networks.*

**Keywords**— Ethernet Flow Control, Ethernet Traffic Shaping, Ethernet Congestion Management

## I. Introduction

The architecture of the local area networks is facing new challenges with the emergence of cloud computing [1], and the deployment of massive data centers [2]. This new computing paradigm allows users to access a virtual network of resources in the cloud that can be called upon to deploy applications on demand. At the same time, the networking research community has moved toward creating similar platforms for experimenting with new networking concepts and architectures [3]. As in cloud computing, these networking testbeds offer a virtual network of resources to the researchers so that they could evaluate their networked systems in large scale.

The creation of these research testbeds and cloud computing platforms has become possible mainly due to the advancement of virtualization techniques that have made separation of the virtual computing resource and the underlying physical resources much easier, and have allowed operation of multiple virtual machines on one physical resource.

Inherent in such shared resource environments is the potential for disruptive interaction among users and hence the need for new techniques to provide network and resource isolation. The Virtualized Application Networking Infrastructure (VANI) [3], [4], developed at University of Toronto, is an example of a networking research testbed that allocates a virtual network of resources to researchers. An important requirement in VANI is to guarantee network access rates and isolation between different experiments. In this paper, we present the Distributed Ethernet Traffic Shaping (DETS) system and its corresponding algorithms designed to provide a guaranteed network access rates in VANI. The DETS system is not only applicable to VANI, but also to the computing clusters and data centers that virtualize and share their resources among different virtual networks. DETS deployment in a cluster or a data center does not require any changes in system hardware, and can be deployed on top of normal computing blades and Ethernet switches.

The primary role of DETS is to control and regulate the traffic sent and received on VLANs. Especially, this is required where more than one virtual machine is working on a physical node, and each has to send and receive a guaranteed rate of traffic on a dedicated VLAN on a shared Ethernet access. Figure 1 shows a sample scenario for DETS. In this sample system, we have five physical nodes (PN) each having two running virtual nodes (VN). All these PNs are connected to an Ethernet network and the VNs running on these PNs require a guaranteed access rate to the Ethernet network. For the sake of simplicity, we show an Ethernet network with just one Ethernet switch, but in general, it is possible to have many switches in a network. In this topology, VNs running on a node are working separately and can only communicate with their peer VNs in other physical nodes.

If $VN11$, $VN12$, $VN13$, and $VN14$ start sending traffic to $VN15$ , they can consume all the available bandwidth on the Ethernet link that connects $PN5$ to the Ethernet

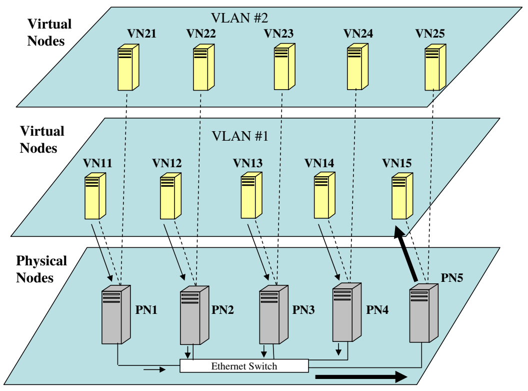

Fig. 1: A system with five nodes and two virtual nodes on each

switch. This may cause problem for traffic sent from nodes $VN21$, $VN22$, $VN23$, $VN24$ to node $VN25$ that shares the Ethernet link with $VN15$. Therefore there is a need for a traffic shaping or rate control to limit the rate that $PN5$ can receive traffic for $VN15$ so that $VN25$ can also receive traffic at a guaranteed rate.

This problem would become very evident and observable if the interfering traffic (traffic for $VN15$) is UDP and the underdog traffic (traffic for VN25) is TCP. The high amount of UDP packets on the link to $PN5$ would virtually disable TCP traffic to $VN25$ as the experimental results in figure 2 show. In the figure, $VN25$ receives the maximum possible TCP rate, if no traffic is sent to node $VN15$. However, as soon as UDP traffic is sent to node $VN15$ (around time 300 in figure 1), TCP rate goes to almost zero until UDP traffic stops (at around time 1000). This experiment shows not only the sensitivity of a TCP flow rate to a competing UDP flow but also it shows the importance of having a traffic shaping and rate control system to guarantee an agreed access rate for different virtual nodes on a physical node that share one Ethernet link. The problem of network performance degradation in virtualized environments has been also studied in [5], and the authors, through measurements on Amazon Elastic Computing services, concluded that virtualization techniques can cause significant throughput instability.

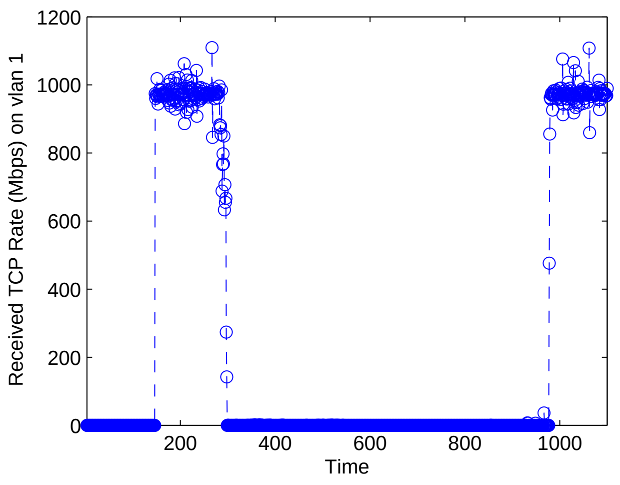

Fig. 2: TCP rate back off due to interfering UDP traffic

Current Ethernet flow control uses PAUSE signals [6]. When multiple ports flood a port, the Ethernet switch sends PAUSE signals back to the flooding ports so that they stop sending for an amount of time specified in the PAUSE message. It has been generally accepted that the pause mechanism in Ethernet flow control is not suitable for solving new challenges facing these networks [2]. To address Ethernet congestion problems, two new IEEE task forces (802.1Qua [7] and 802.1Qbb [7]) have been created. The main approach in these task forces is to do flow control at the level of class of service by marking frames at Ethernet switches. In contrast to these approaches, our proposed system operates at the edge of the Ethernet network on the computing hosts in a cluster or a data center.

We direct interested readers to [2], [8], [9], [10] for a survey on the recent work on Ethernet network congestion control for data centers. The current proposed methods for congestion management entail modifying Ethernet network elements. Moreover, the majority of the proposed systems are Congestion Notification based systems with no explicit rate information [8], [9] that have been shown that have draw backs, such as slow recovery in comparison to explicit rate systems [9].

The salient explicit rate congestion management system, Forward Explicit Congestion Notification (FECN) [9], passes explicit rate from the congestion point to the source point based on the utilization ratio of the congested link. Our system is also an explicit rate system, but differs from FECN in several aspects.

The DETS system is more than just an Ethernet congestion management system. In particular DETS allows setting guaranteed limits on the send and receive on each virtual network, and shapes the traffic so that virtual networks do not interfere with each other's ability to send and receive traffic. Unlike FECN, DETS does not need any change in the current Ethernet equipments, and can be applied in the current computing cluster systems and data centers. Moreover, our system is capable of supporting both fair and weighted fair bandwidth allocation mechanisms. In addition, in allocating rates to the sending nodes, the system considers the available sending capacity of the sending nodes that results in higher throughput.

The DETS operation is seamless to the virtual machines running on the host system, and virtual machines only see the decrease and increase in send and receive traffic rate on certain flows. However, since our system runs on the host system its rate set and measurements periods are limited to the system's timer (about 55ms). DETS also does not explicitly account for in-network congestions.

The organization of this paper is as follows: Section 2 describes our proposed system, identifies key control and measurement points, and presents the DETS protocol. Section 3 presents the DETS system design and it main internal modules. In this section, we also propose four different algorithms developed for DETS. The DETS system performance measurements are presented in section 4, and in section 5, we describe the modification in Ethernet control plane in order to port the DETS system to Ethernet network elements. Finally in section 6, we present concluding remarks and our future work.

## II. Distributed Ethernet Traffic Shaping (DETS) system

The DETS system is designed to control the rate of the traffic generated by each virtual machine according to the total traffic rate at the destination virtual node. DETS controls the sending rate of the traffic in the originating VN before it enters the Ethernet network based on a target rate imposed by the receiving virtual node.

In the VANI system, a virtual LAN is created for the virtual nodes that are in one group, and ”over the top” rate

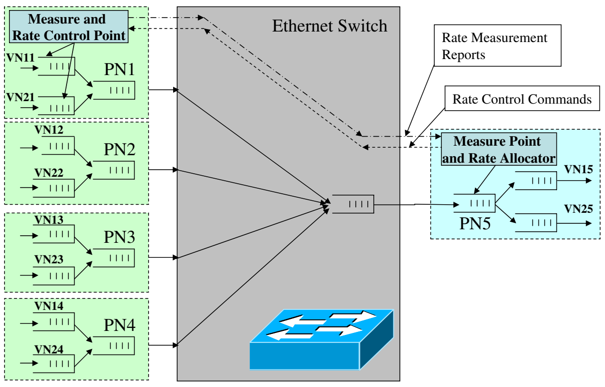

Fig. 3: DETS measurement and rate control points

controller software is run in each of the physical nodes. This software is able to control the rate at which each virtual machine sends traffic to any other virtual machine in that virtual network. The module is also able to measure received traffic to each virtual node, and detect if the received rate limit is violated. If the received rate limit is violated, the receiving node is declared the congested node. The controller then monitors the sent traffic to the congested node and controls its rate at the sending node. This system is depicted in figure 3 which shows the control and measurement points.

Each agent in DETS has two separate modules; a send rate controller, and a receive rate allocator. The send rate controller monitors the sending traffic rate to any other virtual machine that is facing congestion, and reports it to the rate allocator in the congested node (node $PN5$ in example scenario, called the receiving node in the remainder of this document). The rate allocator at the receiving node ($PN5$) allocates a rate to each sending node and sends set-rate commands to the corresponding send rate controller modules in the sending nodes. The send rate controllers apply the received set rate commands (at the set rate control points shown in figure 3) and subsequently the traffic sent to the congested node ($PN5$) will be shaped accordingly.

The DETS system can be implemented in any cluster with any operating system that is able to control the egress Ethernet traffic rate. In the next section, we focus on a cluster of Linux-based computing nodes, and we describe the system design and protocol for deploying DETS in such a cluster.

### A. DETS Protocol

The DETS protocol has five types of messages:

- Traffic Report message, sent from a sending to a receiving node and includes measured rate, current rate limit, and available rate.
- Initialize Traffic Control message, sent from a receiving to a sending node to initialize the traffic controller to the receiving node.
- Set Rate message, sent from a receiving to a sending node and includes the allocated rate that the sending node has been granted.
- Keep Alive message, sent from a receiving to a sending node when the traffic control on the receiving node is active.
- Deactivate Traffic Control message, sent from a receiving to a sending node to deactivate traffic control to that receiving node.

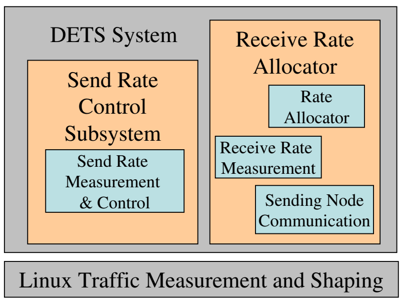
DETS System
Send Rate Control Subsystem
Send Rate Measurement & Control
Receive Rate Allocator
Rate Allocator
Receive Rate Measurement
Sending Node Communication
Linux Traffic Measurement and Shaping

Fig. 4: DETS System Internal Modules

### B. DETS for Linux OS

In Linux, traffic shaping can be done on egress and ingress traffic. The main command for performing traffic shaping is 'tc' command [11]. This command can operate on a virtual interface (serving a VLAN), and can be also used for measuring the send and receive rates. The shaping in our system is done in the Linux hosts, and it is seamless to the virtual machines running on them.

## III. DETS System Design

Figure 4 shows the design of DETS. In the send rate control module, there is one state machine for each receiving node. Also, there are two internal sub modules in the receiving rate allocator module. The first module is responsible for communicating with the sending nodes, and the second module allocates the rates to the sending nodes.

### A. Rate Allocator Module

The core part of the DETS system is the rate allocator module that allocates the sending rate to each sending node. The rate allocator module utilizes a Rate Allocation Algorithm (RAA) to determine the rate at which each sending node can send traffic to the receiving node.

In RAA design, we need to consider that the measurements in the send rate control modules are capped by the rate set by RAA. To better explain this limitation and its implication on algorithm design we use an example scenario. Assume that in figure 3, the system in a steady state with four virtual nodes ($VN11$ to $VN14$) sending traffic to VN15 with rates (80, 80, 20, 20)Mbps respectively. At this point, if VN11 stops sending traffic to VN15 the rate allocator algorithm may reallocate the vacant rate to other nodes. However, since there are no measurements for sending rate above the rate limits, the RAA needs a mechanism to probe $VN12$ to $VN14$ to see if the sending nodes need to send more traffic or not. Without a probing mechanism, RAA would allocate rates to a node that might

```
input : Active nodes list and their requested rate and send capacity
output: Calculates granted rate to each node

1) Inflate requested rate of the nodes that fully use their allocated rate by 10%;
2) Calculate the total requested rate;
3) Calculate the ratio of increase and decrease the requested rate based on the available rate;
$ratio \leftarrow totalReqRate/totalAvailableRate$;
4) while There are unallocated rate and nodes with sending capacity do
    $grantRate[i] \leftarrow min(reqRate[i] * ratio, maxRate[i])$;
    if $reqRate[i] * ratio > maxRate[i]$ then
        fairly distribute extra rate among other nodes;
    end
end
```

**Algorithm 1:** RAASlowProbe

not need the extra allocated rate and the available bandwidth would be wasted.

The probing mechanism allows us to provide fairness in rate allocation to virtual nodes. Assume that in the above example, all nodes have similar importance, and have equal amount of traffic to send to VN15, so the above allocated rate is not fair since two of the virtual nodes have been allocated rates (80 Mbps to each) that are much more than the rates allocated to the other two nodes. If $VN13$ and $VN14$ had more traffic to send this rate allocation is unfair. In this case, the probing mechanism in RAA starts probing nodes with lower allocated rates to see if they have more traffic to send, and whether they need more allocated rate.

The probing mechanism in RAA is done through gradual increase and decrease in rate allocations to different nodes and monitoring the increase and decrease in rate measurements. The probing mechanism may reduce bandwidth utilization, but this might be acceptable in order to overcome the above mentioned problem.

Another important factor in RAA design is to consider the available traffic sending capacity in the sending nodes in rate allocation. Assume that in figure 3, $VN14$ is sending 20 Mbps to $VN15$, and 80 Mbps to $VN12$, and its total send limit is 100 Mbps. Therefore, $VN14$ cannot send any more traffic to $VN15$. Therefore, the rate allocator algorithm in $PN5$ should consider the available sending capacity of the sending nodes in its rate allocation.

There are a number of possible allocation algorithms that can be used in this system. Next, we propose four of these rate allocation algorithms: Fair Share algorithm (RAA-FS); Slow Probe algorithm (RAA-SP); Fast Probe algorithm (RAA-FP), and Forward Explicit algorithm (RAA-FE).

The fair share algorithm (RAA-FS) calculates a fair rate by dividing the receiving rate limit by the number of sending nodes that have traffic to send, and allocates that fair share to each of the active sending nodes. This algorithm is suitable for the cases where the sending nodes need to be treated similarly in the rate allocation process, independent of the amount of required traffic.

```
input : Active nodes list and their requested rate and send capacity
output: Calculates granted rate to each node

1) Execute Slow Probe algorithm;
$grantRate \leftarrow$ RAASlowProbe();
2) Sort all nodes that fully utilized their allocated rate according to their granted rate, and calculate the mean of the granted rate to them;
3) while pick a node with highest rate above mean rate(upper) do
    while pick a node with lowest rate below mean rate(lower) do
        Multiply the rate of lower node by $d$ and deduce the increase from higher node, considering lower node send capacity;
        if upper node new rate goes below mean then
            average lower and upper rate and assign avg rate to both;
        end
    end
end
```

**Algorithm 2:** RAAFastProbe

In this rate allocation mechanism, if the calculated fair rate is more than the sending capacity of a sending node, the extra rate is fairly distributed among other sending nodes with available sending capacity. This algorithm is oblivious to the difference in rate requested by each active node, and does not perform any probing to see if the nodes have more traffic to send or not. Although RAA-FS is fair but it might result to bandwidth underutilization, since some sending nodes might not need all of the allocated rate.

The second algorithm, slow probe algorithm (RAA-SP), allocates rates to the sending nodes based on the rate measurement reports received from their send rate control modules. The algorithm identifies the nodes that are fully utilizing their allocated rate, and inflates their rate request by a percentage (for example 10%) to give them an opportunity to increase their rate realtive to other sending nodes that are not using their allocated rate. RAA-SP then calculates the total requested rates and allocates a portion of the available bandwidth to each node. This portion is calculated based on the inflated request rate and the receive rate limit as presented in this algorithm's pseudo code (Algorithm 1).

RAA-SP gradually probes the sending nodes that are fully utilizing their allocated rate, and gives them a better chance of getting more allocated rate. RAA-SP, however, does not address the fairness problem, since it does not reallocate the rate from the high rate allocated nodes to the low rate nodes.

The third algorithm, fast probe RAA (RAA-FP), extends the slow probe algorithm by reallocating the sending rates from the higher rate allocated nodes to the lower rate allocated nodes. In contrast to the two previous algorithms, RAA-FP addresses both fairness and bandwidth utilization concerns. This algorithm sorts the nodes that fully utilize their allocated rate and calculate the mean allocated rate to these nodes (shown in the pseudo code presented in Algorithm 2). RAA-FP then picks the nodes with the highest allocated rate, and the lowest allocated rate. RAA-FP multiplies the rate allocated to the lowest rate allocated node by a parameter ($d > 1$) and deducts that extra allocated rate from the node with highest allocated rate if the resulting deducted rate does not go below the mean allocated rate. Otherwise, it takes an average between the highest and lowest rate allocated nodes, and allocates this average rate to both of them. This change in the allocated rate is done considering the free sending capacity of the node with lower allocated rate. This operation is repeated on the next two nodes with the next highest and lowest allocated rates until all allocated rates to the fully utilizing nodes get revised.

Our performance evaluations show that the fast probe rate allocation algorithm (RAA-FP) is able to achieve probing algorithm goals rather quickly, since it gives more opportunity to nodes that are fully utilizing their allocated rate to send more traffic. Moreover, it achieves better fairness in rate allocation since it reduces the gap between the nodes with high allocated rates and nodes with low allocated rates. The choice of parameter $d$ controls the trade off between the fairness and bandwidth utilization. A small $d$ value results in more bandwidth utilization but lowers fairness in rate allocations. On the other hand, A choice of large $d$ results in lower bandwidth utilization in exchange of higher fairness in rate allocation.

The fourth algorithm is inspired by the FERA algorithm introduced in [9] for FECN-based Ethernet congestion management. This algorithm is designed to enable comparison between a DETS-based rate allocation system and a FECN-based system. The essence of FERA is to control the queue length of an outgoing Ethernet switch port by assigning a fair share rate to flows passing through that port. This algorithm uses a linear (or a hyperbolic) control function to adjust the allocated (fair) rate to achieve a target level on queue length ($Q_{eq}$).

We modified this algorithm to arrive at a target receiving rate at the receiving node. This algorithm (called RAA-FE) calculates a fair rate ($r_{i+1}$) at $(i+1)th$ interval, based on the $r_i$ value at $i$th interval, and a control function $f(r) = 1 - k * \frac{r-R_t}{R_t}$ in which $k$ is a constant, $r$ is the measured receiving rate, and $R_t$ is the target rate.

DETS sends back the calculated rates to the sending nodes, and the sending nodes apply the rates to their rate controller modules. Compared to the previous algorithms, this algorithm does not require the rate measurements at the sending nodes, and does not support weighted fair allocation.

In the original FERA, intervals are as low as 1 ms, but in DETS intervals are about 55 ms. Therefore, the rate regulations are done every 55 ms that makes rate convergence a challenge for this algorithm. Although the linear control function leads to a faster convergence time compared to the hyperbolic function, but as our experiments show, RAA-FE takes about 40 intervals ($> 2$s) to converge to the fair rate. The analytical results show this slow convergence as well [9]. This is mainly because this algorithm does not include the sending rate measurements.

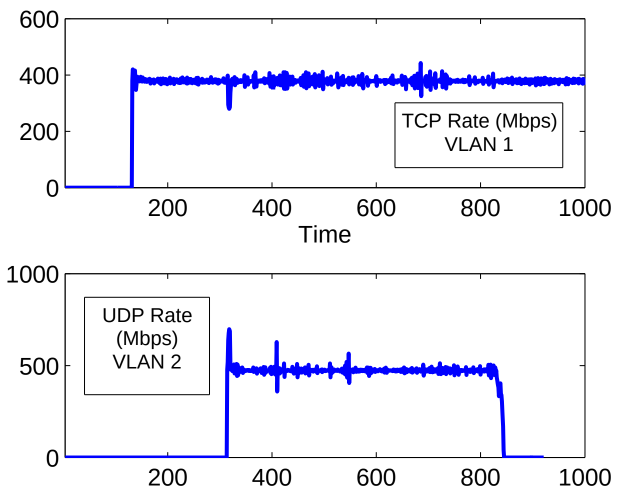

Fig. 5: DETS performance evaluations for system shown in Figure 1

## IV. Performance Evaluations

In this section, we first show that DETS can achieve isolation between virtual LANs. We implemented the DETS system in C++ and deployed it on 11 nodes with 1GE Ethernet connections in a computing cluster, and we created two VLANs on the Ethernet switches. As in our VANI processing virtualization service [4], we used Linux vServer technology for virtualization and deployed two virtual nodes on each physical server. One virtual node in a physical node is connected to VLAN 1, and the other one is connected to VLAN 2. This setting is similar to the one depicted in figure 1, except that we used eleven physical nodes instead of five nodes.

We set the send and receive limit rate for all virtual nodes in the first VLAN to 400 Mbps, and in the second VLAN to 500 Mbps, and we used the fast probe rate allocation algorithm on both VLANs with parameter $d = 2$. We started sending TCP traffic from 10 nodes to one node. We expect that DETS control the rate that the receiving node receives traffic, and limit it to 400 Mpbs. We also expect that if the nodes in the second VLAN start sending UDP traffic to the receiving node, the TCP flows destined to that machine don't get overwhelmed with the interfering UDP traffic.

Our results (presented in figure 5) show that DETS is able to achieve both goals. In this figure, the rate measurements are shown in every time unit (every 55 ms). As can be seen, when all nodes in the second VLAN simultaneously start sending UDP traffic to the receiving node (around time unit 320 in figure 5), momentarily TCP traffic on the first VLAN gets disrupted, and it takes two time units for the control algorithm to receive the measurements and make the decision and apply the limits on the sending nodes. After

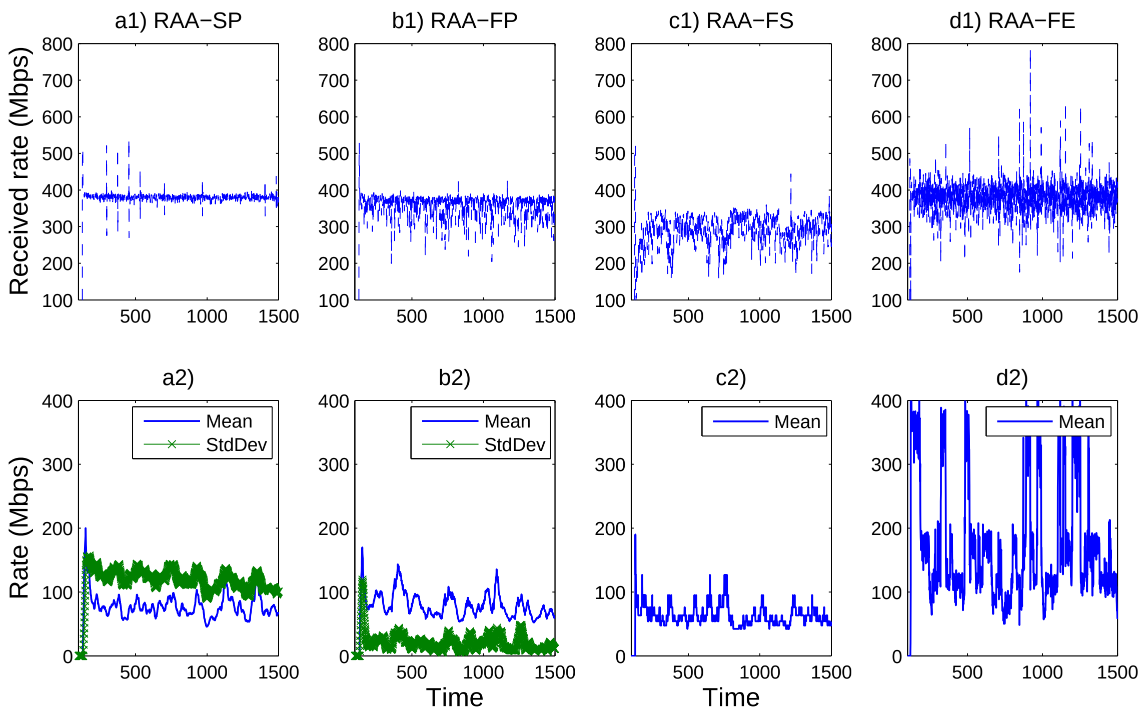

Fig. 6: Performance evaluation of the four rate allocation algorithms a) RAA-SlowProbe b) RAA-FastProbe c) RAA-FairShare d) RAA-ForwardExplicit

this short transient period, TCP traffic is able to bounce back quickly, and continue sending information at the limit rate which is 400 Mbps.

We also evaluated and compared the performance of the four allocation algorithms. To do so, we set up a VLAN with 10 virtual nodes sending a mix of UDP and TCP traffic to one virtual node, and we monitored the received traffic on the receiving node. We also limited the peak rate of three of the sending nodes to a low limit (to 20Mbps). This helps us better compare the performance of the proposed algorithms.

We developed an on/off burst traffic generator that is able to generate a burst of UDP or TCP traffic for a random period between $0$ and $T$, and stops sending traffic for another random period between $0$ and $T$. We used various values for $T$ ranging from $0.5s$ to $10s$ on different nodes. This traffic generator enables DETS performance evaluation under time varying and bursty UDP and TCP traffic.

Figures 6(a1, b1, c1, d1) show the received rate measurements on the receiving node for all algorithms for the period of 82 seconds (1500 time units). Figures 6(a2, b2) show the measured mean and standard deviation of the allocated rate to the nodes by the slow probe and fast probe ($d = 2$) algorithms, respectively. Figures 6(c2, d2) show the mean value of the allocated rate by RAA-FS, and RAA-FE. The fluctuations in the received rate measurements are due to the on-off nature of the generated traffic.

It can be seen that the fast probe and the slow probe algorithms achieve better utilization of the received bandwidth compared to the fair share algorithm, especially since some of the nodes have less sending capacity compared to the other nodes. As it was expected, the slow probe algorithm outperforms the fast probe algorithm in term of its receiving bandwidth utilization. However, the fast probe algorithm is able to achieve a low standard deviation between different flows coming from different virtual nodes compared to the slow probe algorithm.

The RAA-FE algorithm performs poorly compared to the other algorithms and has a slow convergence rate, and it has difficulty stabilizing. This is mainly because of the fluctuations in the generated traffic. RAA-FE also does not consider the sending rate measurements, and does not have a probing mechanism.

In general, the fast probe algorithm is better than other algorithms if weighted fairness is required, but if a user needs fairness in rate allocation the fair allocation schema can be picked. The slow probe algorithm is for the cases where the user wants to increase the bandwidth utilization in expense of fairness, and does not want a sudden change of a traffic flow rate and prefers a slow change. In DETS, it is possible to have different rate allocation algorithms running on different virtual networks, as long as algorithms satisfy the network isolation requirement. This allows users to pick an algorithm that suits their needs.

## V. Modifications to Ethernet Control Plane

Here we discuss inclusion of DETS protocol in the Ethernet control plane so that Ethernet switching equipments can perform DETS operations even without (or with minimum) help from hosts attached to the Ethernet network.

We propose that in an Ethernet network, the distributed modules in the DETS system be embedded in Ethernet switches, and DETS messages be added to the Ethernet control messages. To do so, controlling traffic to a receiving

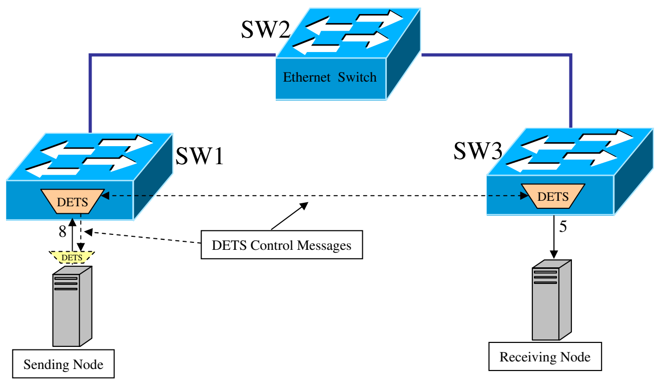

Fig. 7: DETS in Ethernet control plane

node has to be done on ingress ports by the edge Ethernet switches. These messages could be added to the MAC Control type of Ethernet frames (EtherType = 0x8808) as specified in IEEE 802.3 family of specifications [6]. The only message currently defined in this type of frame is the PAUSE message (opcode = 0x0001). The DETS messages can use other free opcodes in this frame type. These messages have to be in VLAN-tagged frames, since DETS is designed to control the rate on VLANs.

Figure 7 shows an Ethernet network equipped with DETS. The rate allocator module operates on the receiving port of an edge Ethernet switch (SW3,port number 5) and the send rate control module and the traffic shaper operates on the sending port of originating edge Ethernet switch (SW1,port number 8). The set rate messages are sent from the receiving port to the sending port. The sending port applies the allocated rate to the sent traffic, and can forward the rate control messages to the sending host in case it (or its NIC) is able to do the traffic shaping.

## VI. Conclusion and Future Work

We presented a distributed system for traffic shaping in Ethernet networks. The DETS system is required where there is a host node connected to several virtual local area networks, and the sending and receiving traffic rate on each of these virtual networks has to be guaranteed and controlled. Without this control, an excess of received traffic on one of these virtual networks could disturb other virtual networks ability to receive traffic in a guaranteed rate. We presented a design and a protocol for the DETS system. Moreover, we proposed four different algorithms for rate allocation in DETS, and through experimental performance evaluations showed DETS effectiveness in an example network.

We also proposed modifications to the Ethernet control plane so that DETS can be natively supported by Ethernet networking elements. We intend to further explore these modifications and develop proof of concept Ethernet switches with this capability using the hardware resources developed for VANI.

## References

[1] Cloud Computing Definition, National Institute of Standards and Technology, Version 15, 2006. Available at http://csrc.nist.gov/groups/SNS/cloud-computing/index.html.

[2] Data center evolution: A tutorial on state of the art, issues, and challenges. *Computer Networks*, 53(17):2939 – 2965, December 2009.

[3] Bannazadeh H., Leon-Garcia A., and et. al. Virtualized Application Networking Infrastructure. In *Proc. of the 6th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, to appear*, Berlin, Germany, May 2010.

[4] Redmond K., Bannazadeh H., Leon-Garcia A., and Chow P. Development of a Virtualized Application Networking Infrastructure Node. In *Proc. of the 3rd IEEE Workshop on Enabling the Future Service-Oriented Internet*, Honolulu, Hawaii, December 2009.

[5] Guohui Wang and T. S. Eugene Ng. The impact of virtualization on network performance of amazon ec2 data center. In *Proceedings of the 29th IEEE Conference on Computer Communications, INFOCOM 2010*, San Diego, CA, March 2010.

[6] IEEE 802.3x-1997, Local and Metropolitan Area Networks: Specification for 802.3 Full Duplex Operation, 1997. Available at http://standards.ieee.org.

[7] IEEE 802.1au, Virtual Bridged Local Area Networks Amendment Congestion Notification. Available at www.ieee802.org/1/pages/802.1au.html.

[8] M. Alizadeh, B. Atikoglu, A. Kabbani, A. Lakshmikantha, Rong Pan, B. Prabhakar, and M. Seaman. Data center transport mechanisms: Congestion control theory and ieee standardization. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 1270–1277, Sept. 2008.

[9] Jinjing Jiang, R. Jain, and Chakchai So-In. An explicit rate control framework for lossless ethernet operation. In *Communications, 2008. ICC '08. IEEE International Conference on*, pages 5914–5918, May 2008.

[10] Gary McAlpine, Manoj Wadekar, Tanmay Gupta, Alan Crouch, and Don Newell. An architecture for congestion management in ethernet clusters. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium - Workshop 9*, page 211.1, 2005.

[11] Linux Advanced Routing and Traffic Control. Available at http://lartc.org/.