Emulation of Differentiated Services using VNUML

CS 680 Course Project

Vijay Gabale (07305004), Sagar Bijwe (07305023), Manish Kumar (07405701)
Computer Science And Engineering Department,
Indian Institute of Technology, Bombay,India
Course Instructor : Prof. Anirudha Sahoo

May 1, 2008

Abstract

Differentiated Services (DiffServ) has been proposed by IETF as a scalable QoS solution for the next generation Internet as an alternative to Integrated Services (IntServ). It has been developed for relatively simple, coarse methods of providing different levels of service for Internet traffic. The main idea is to divide traffic into a small number of classes and allocate resources on a per class basis. The Core of a diffserv network then distinguishes between small number of forwarding classes rather than individual flows as in Integrated Services. Through this project, we have implemented various components of DiffServ like classification, marking, dropping and scheduling. We have further analysed how a DiffServ enabled network behaves under varying load conditions and in terms of real time flows. For the emulation and analysis purpose, we have used VNUML (Virtual Network User Mode Linux) tool.

For classification purpose, we use multi-trie approach to match IP address according to SLAs (Service Level Agreements) between service provider and customers. To do marking, we use two rate three colour marker token bucke approach and for scheduling, we employ priority scheduling with flows constrained by token buckets. Through the experiments performed, we come up with following key conclusions : (a)(b)(c)(d)

1 Introduction

To mitigate the disadvantages of having per flow reservations at every intermediate node, which requires tremendous state maintenance in the core, and having complex scheduling in IntServ, DiffServ [1] was proposed.

The basic approach of DiffServ is as follow : Traffic is divided into a small number of groups called forwarding classes. The forwarding class that a packet belongs to is encoded into a field in the IP packet header. Each forwarding class represents a predefined forwarding treatment in terms of drop priority and bandwidth allocation. This scheme achieves scalability by implementing traffic classification and conditioning functions at network boundary nodes. Classification involves mapping packets to different forwarding classes whereas conditioning consists of checking whether traffic flows meet the service agreement and dropping nonconformant packets. Interior nodes forward packets based solely on the forwarding class. Thus, the resource allocation is made for aggregated traffic rather than individual flows. Still, the performance assurance to individual flows in a forwarding class is provided through prioritization and provisioning rather than per-flow reservation. Also, DiffServ defines forwarding behaviors and not end-to-end services. Further the service assurance is based on SLAs and not dynamic signaling as in IntServ using RSVP (Resource Reservation Protocol). Also, the focus has been made on a single domain rather than end to end resource reservation.

This motivates us to implemente these theoretical concepts into practice and to get hands dirty with scheduling, token bucket, IP address prefix matching schemes. Some of the interesting questions that we ask are

- What is order of improvement achieved by DS enabled service in terms of delay, jitter and throughput?

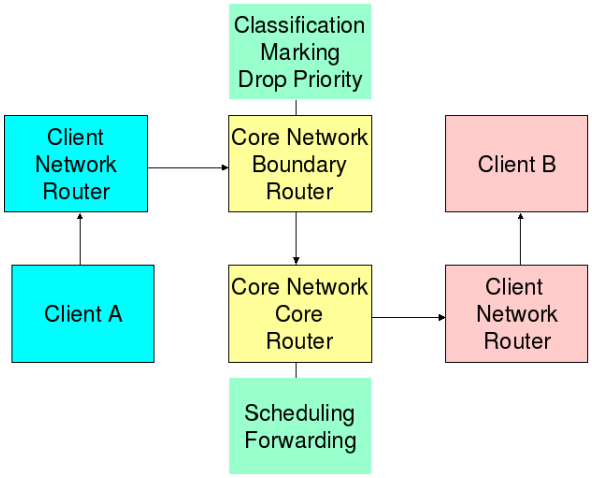

Figure 1: Architecture of DiffServ enabled network

- How do real time flows behave in DS enabled network?

The main observations that we come up with are as follows

The rest of the report is organized as follows : in section 2, we further delve into the DiffServ and exemplify our architecture. In section 3, we explain the multi-trie scheme for classification, the two rate three colour marking scheme for assigning drop priorities to packets and priority scheduling with token buckets. In section 4, we briefly explain the usage of VNUML. In section 5, we present the various experiments performed and the respective results. Here we particularly compare DiffServ enabled and DiffServ non enabled statistics for the same traffic pattern and same network topology. In section 6, we make some quick comments regarding our observations. Finally we give future scope and conclusion in section 7 and 8 respectively.

2 DiffServ Architecture

The figure 1 exemplifies one particular flow in the DiffServ network. The flow emanates from the client network and passes through the boundary router. Then it enters the DiffServ enabled network through boundary router. The boundary router does the work of classification and marking the packets. The classifier divides an incoming packet stream into multiple groups based on predefined rules. The two basic types of classifiers : Behavior aggregate (BA) and Multifield (MF). BA classifier selects packets based solely on DSCP value in the packet header and it is used when DSCP has been set (marked)

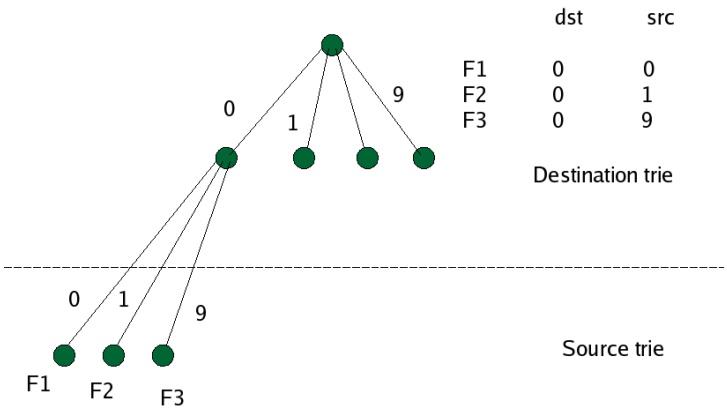

Figure 2: Multi-trie approach of flow classification

before the packet reaches the classifier. Whereas MF classifier uses a combination of one or more fields of the five-tuple (src addr, src port, dest addr, dest port, proto ID) in the packet header for classification. We have used MF classifier and the classification is based on multi-trie approach with the help of SLA. Then the packet is forwarded through core routers who employ priority scheduling with token buckets for each flows. The DSCP values can be used to make this service differentiation. The corerounter only does the work of forwarding the packets based on this forwarding class. Finally the traffic reaches to destination client B.

3 Implementation Details

This section explains the multi-trie approach of address classification, two rate three colour approach to mark dropping priorities and priority scheduling with token bucket constrained flows.

We have defined our own custom packet header format at the application layer and use this header to do the work mentioned in the above paragraph. This header consists of Source address, Destination address, Source port, Destination port, Type of Service (forwarding class) and Drop Priority.

The multi-trie approach is shown in the figure 2 where the table entries indicate the subnet addresses of the customer networks.

Here, first the destination subnet address is extracted from the packet destination field. This is then matched with the first level of the tree after which the source subnet address is used to arrive at the forwarding class for this particular flow. The Type of Service field is set in this way.

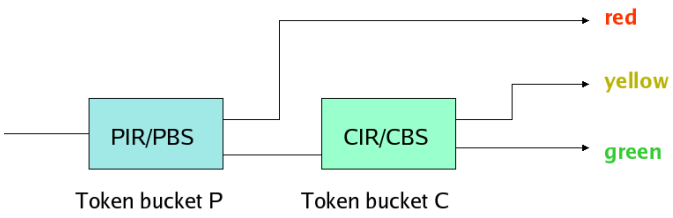

Figure 3: Two rate three colour marker

Next, the packet dropping priority is marked according to two rate three colour marker. The metering is based on the token bucket bucket size and rate and the non conforming flows are as red, yello and green as shown in following figure 3. This two rate three colour marker has four configurable params: peak rate, peak burst size, committed rate and committed burst size. Each packet passes through two token buckets as shown in figure. Color is assigned to the packet depending on the results of test by the two token buckets: packet is marked green if it passes the tests of both token buckets P and C, packet is marked yellow if it passes the P bucket but not C bucket and packet marked red if it fails both tests. The red packets are dropped first in case congestion, then yellow and then green packets in the worst case if the congestion persists.

For forwarding at the core routers the priority scheduling policy is used. The queues are constrained by the token buckets to rate limit the traffic. The EF (forwarding class 71 in our emulation) queue is given the highest priority to allow low loss, low delay and high bandwidth traffic. Then there are flows with class 72 to 77 in the order of their priority from highest to lowest.

4 VNUML usage

VNUML (Virtual Network User Mode Linux) [2] is an open-source general purpose virtualization tool designed to quickly define and test complex network simulation scenarios based on the User Mode Linux (UML) virtualization software. VNUML is a useful tool that can be used to simulate general Linux based network scenarios. It is aimed to help in testing network applications and services over complex testbeds made of several nodes (even tenths) and networks inside one Linux machine, without involving the investment and management complexity needed to create them using real equipment. VNUML tool is made of two main components: the VNUML language used for describing simulations in XML; and the

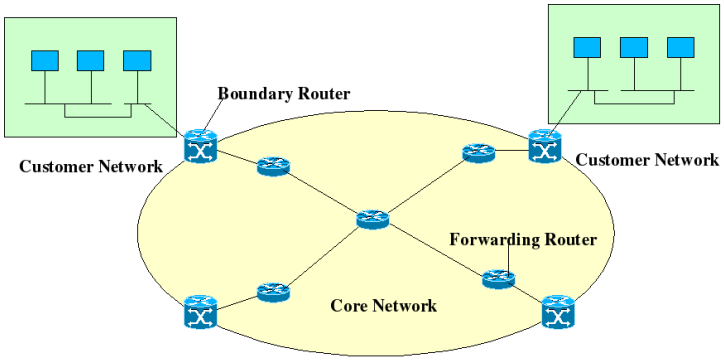

Figure 4: Network topology for the experiments

interpreter of the language (vnumlparser.pl), that builds and manages the simulation, hiding all UML complex details to the user.

For the network shown in figure 4, we wrote xml file for VNUML to boot up the network with requisite programs that perform tasks for core boundary router, core router etc. The command sequence to follow to run the emulation is as follows:

1. vnumlparser.pl -t project.xml -v : This command builds the virtual network topology described in project.xml file and boots all the virtual machines defined inside it.

2. vnumlparser.pl -d project.xml -v : This command is used for releasing virtual machine scenario. It sends the Ctrl-Alt-Del sequence to every virtual machine.

There is also provision to start programs to run in each virtual machine as we start the emulation. We make use of this utility to start the emulation as we start the vnuml virtual machines.

5 Experimental Details

With this arrangment in place, we performed two experiments. One where there are two real time flows with DiffServ enabled and disabled network and compare the performace in terms of delay, jitter and throughput. In second experiment we vary the token generation rate for the two flows and observe their behaviour.

5.1 Comparison DiffServ enabled and disabled network

Here, we have two flows each generating packets after 20ms (thus we simulate two real time flows). The SLA is

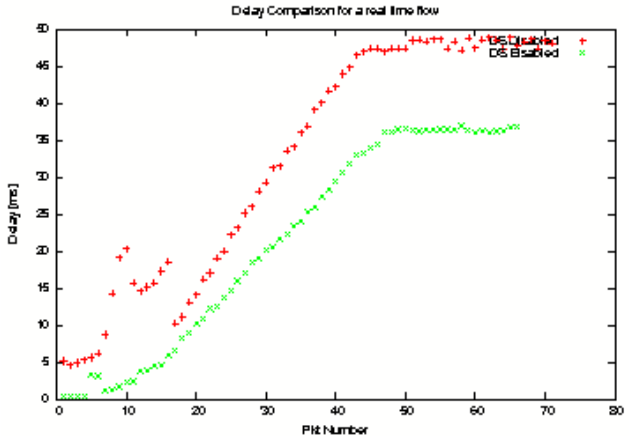

Figure 5: Comparison of delay

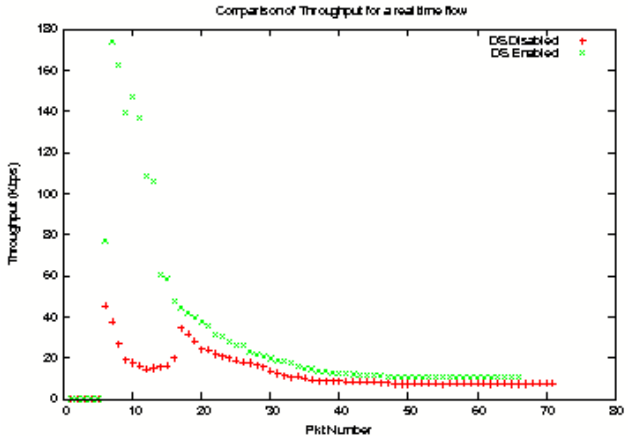

Figure 7: Comparison of throughput

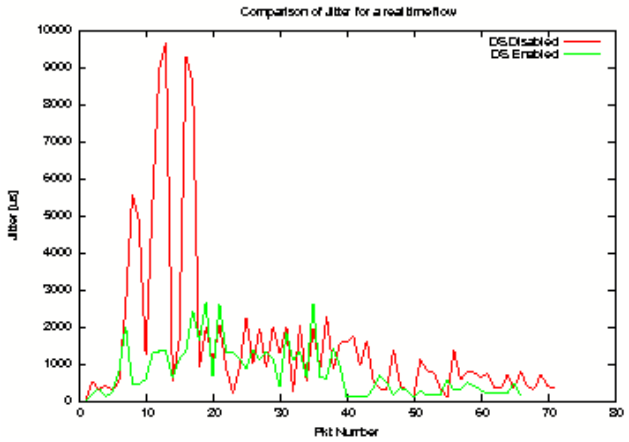

Figure 6: Comparison of jitter

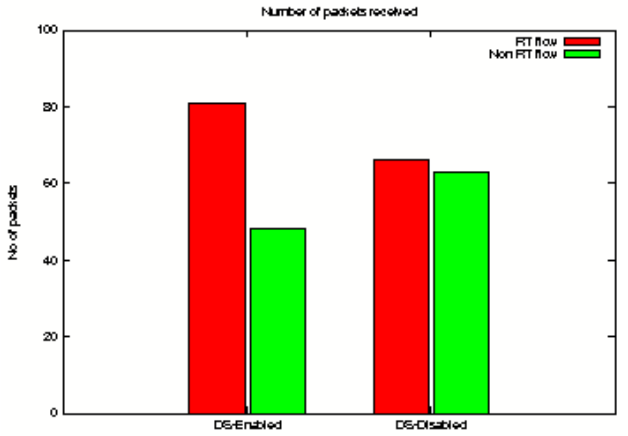

Figure 8: Number of packets received

in place and one flow has been assigned to Expedited Forwarding class (EF). The other is assigned slightly lower priority. First we disable DiffServ in the network and run the two flows. Next we enable DiffServ in the network. We measure and compare the delay, jitter and througthput of two flows. These experiments were performed with the intention of quantifying the performace improved of DiffServ enabled network.

Figure 5, 6 and 7 show the results. For all graphs, we observe that, the EF flow performs far better than the other normal priority flow. We surmise that the slight variations in delay in between are due to processing overheads in VNUML. For delay graph, the two lines are for one flow only. In DiffServ disabled network, it gets normal priority as other flows whereas in DiffServ enabled network, it gets higher priority and thus lower delay. Same way, the jitter is very much unstable and varies between highs and lows in case of DiffServ disabled network but due to slightly predictable nature of EF service, the jitter is very much controllable in case DiffServ enabled network. We also measured the throughput and found that throughput for the same flow in case of DiffServ enabled network was greater than disabled.

For 100 packets sent, we also measured the number of packets received and lost for the two flows in DiffServ enabled and disabled network. Figure 8 and 9 show the results. We can see that for DiffServ enabled network, for the EF flow (flow 1 in graphs), the number of packets received far exceed that of other flow and in similar ways the number of packets dropped are far lower. This proves the adequacy of DiffServ network to real time flow.

5.2 Behaviour of flows with respect token bucket parameters

Next, we kept the token generation parameters for the two flows as follows: for one experiment same (2:2 ratio) and for second experiment in the 3:1 ratio. The network set up and forwarding class assignment was kept same. So we expect that flow 1 to get better service than flow

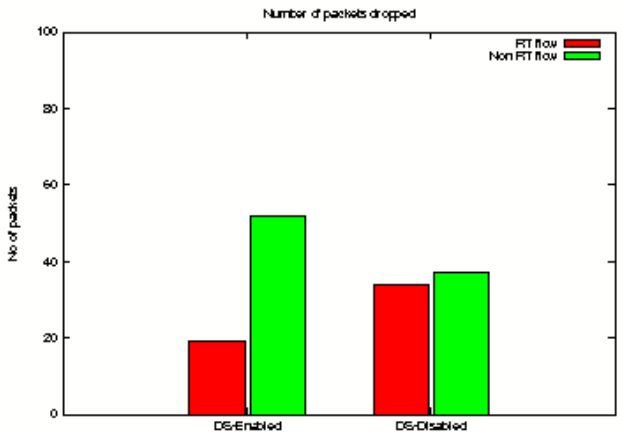

Figure 9: Number of packets dropped

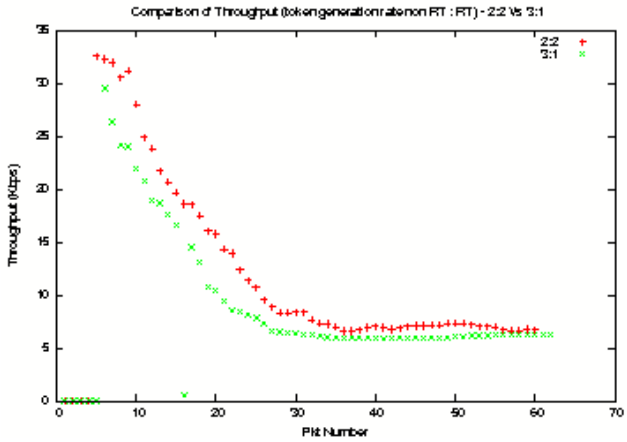

Figure 10: Comparsion of delay

2. These experiments were performed with the intention that, since flows are constrained by token bucket parameters, how do flows behave when we change these parameters. Figure 10 and 11 show the results of the experiments in terms of delay and throughput. The comparison uncovers that the flow with more token generation rate performs better in terms of delay and throughput. Thus it is crucial to set the token bucket parameters appropriately to give proper intended service to flows.

6 Observations

Through the above experiments we make following observations

- The DiffServ enabled network improves performance of higher priority flows. The service given to one of the two real time flows in the experiment was quite predictable. This is also in accordance with that DiffServ doen not guarantee any end to end delay but provides sufficient resources to have improvement as compared to other flows.

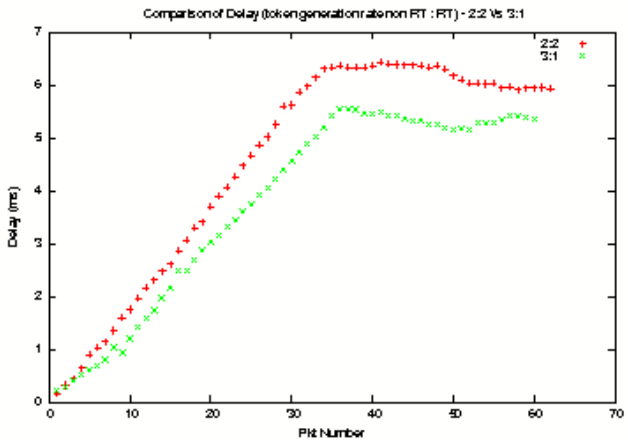

Figure 11: Comparsion of throughput

- The performance improvement was about 25 percent and in terms of packets dropped it was around 27 percent.

- As we increased the token generation rate of one of the flows, more number of packets of the flow got admitted which in turn resulted lower number of packets getting dropped, lesser delay and higher throughput.

We would further like to note some points regarding VNUML usage.

- We had two laptops, one with 2GB RAM and the other with 256MB RAM. The processor was same with specification of 1.46GHz. We observed that VNUML far outperforms in terms of speed of popping windows and processing. The readings taken are on 2GB RAM laptop since VNUML crumbled when we increased network size and number of flows on 256MB RAM.

- The other reason why we stuck to only two flows that, the performance used to severely get degraded when we increased number of flows beyond two flows.

- On 256MB laptop, we sometimes used to get Operating System memory bound exceeded message by VNUML. We suspect there is bug in VNUML implementation which we uncovered when we increased the size of network and number of flows.

7 Future Scope

We performed the experiments with two flows. We would like to extend this scenario to more number of flows and are keen to see how the flows behave. Our gut feeling is that the observations that we have made will remain same though.

Though we considered fairly large network, but we would certainly like to expand this scenario with much larger network and more number of flows to effectively quantify the behaviour of flows under DiffServ enabled network.

8 Conclusion

We implemented various mechanisms for classifcation, marking and forwarding for DiffServ. We emulated a fairly intricate network in VNUML. We performed experiments with two flows by varying the forwarding class assignment. We observe that the DiffServ enabled network improves performance of higher priority flows. The service given to one of the two real time flows in the experiment was quite predictable. This is also in accordance with that DiffServ doen not guarantee any end to end delay but provides sufficient resources to have improvement as compared to other flows. The performance improvement was about 25 percent and in terms of packets dropped it was around 27 percent. The experiments have proved that DiffServ is capable of protecting EF flow in entire network with out setting any complex protocol or mechanism to co-ordinate routers. As we increased the token generation rate of one of the flows, more number of packets of the flow got admitted which in turn resulted lower number of packets getting dropped, lesser delay and higher throughput. We also documented our experiences with using VNUML for using emulation.

References

[1] An Architecture for Differentiated Services. http://www.ietf.org/rfc/rfc2475.txt.

[2] Virtual Network User Mode Linux. http://www.dit.upm.es/vnumlwiki/.