Practical Considerations for Enabling a srTCM Behavior in Opnet Modeler

Pana Flavius and Put Ferdi
Leuven Institute for Research in Information Systems, Katholieke Universiteit Leuven, Leuven, Belgium

Keywords: Qos, Diffserv, srTCM, Simulation, OPNET.

Abstract: Different Quality of Service (QoS) mechanisms have been proposed over time. Differentiated Services (DiffServ) represents one of the main QoS mechanisms developed, and is based on a strategy of traffic differentiation. Introduced in order to be used within DiffServ, the single rate Three Color Marker (srTCM) represents a policer which meters the IP packet stream and marks the traffic with different drop probabilities. This paper presents the technical aspects of implementing the srTCM in one of the most important network simulation tools on the market, OPNET Modeller. Practical considerations and a test case of the proposed implementation are presented.

1 INTRODUCTION

Differentiated Services (DiffServ) represents a scalable option for enabling QoS on the Internet. The basic approach proposed by the DiffServ mechanism is the differentiation of traffic based on traffic classes (Blake, et al., 1998). These classes are identified by the use of the Differentiated Service Code Points (DSCP) introduced in (Nichols, Blake, Baker and Black, 1998). Defined as the "externally observable forwarding behavior applied to a DiffServ node", the forwarding path behavior is identified as the Per Hop Behavior (PHB). Using DSCP, DiffServ classifies each packet to a PHB at every network node from the sender to the receiver host.

Three important types of forwarding behavior or PHB have been standardized over time. These are the Default (DF) PHB, the Assured Forwarding (AF) PHB and the Expedited Forwarding (EF) PHB.

The AF PHB is defined in four different PHB groups with three PHBs in each group (Heinanen, Baker, Weiss and Wroclawski, 1999). The PHB groups are named AF classes and the three PHBs defined in a class represent drop precedence levels for that particular class. The introduction of AF classes gives service providers the possibility to differentiate between the offered services.

The Internet Engineering Task Force (IETF) introduces also a series of specific implementers like traffic markers and traffic shapers, in order to support the functional requests of the PHBs.

Three different types of markers are introduced: single rate Three Color Marker (srTCM) in (Heinanen and Guerin, 1999a), two rate Three Color Marker (trTCM) by (Heinanen and Guerin, 1999b) and Time Sliding Window Three Color Marker (TSWTCM) in (Fang, Seddigh and Nandy, 2000). All the presented markers were intended to mark packets that are treated by the AF PHB. Each of them meters a traffic stream and marks packets to be green, yellow or red. In the specific case of the AF PHB, the color is coded as the drop precedence of the packet. The main difference between the proposed markers is the specific use of rates and parameters in order to mark the packets.

OPNET Modeler represents nowadays one of the most used simulation tools for network researchers (OPNET, 2012). OPNET currently supports the DiffServ mechanism, but it does not have an implementation of the three markers previously presented. Because of this drawback, some broader simulation scenarios, like the examples proposed by (Babiarz, Chan and Baker, 2006) could not be analyzed.

The current work presents some practical considerations on how to implement the srTCM behavior in OPNET Modeler. Illustrated are also some concepts that will further help researchers to implement other types of markers, for example the trTCM.

Flavius P. and Ferdi P..
Practical Considerations for Enabling a srTCM Behavior in Opnet Modeler.
DOI: 10.5220/0004052804490452
In *Proceedings of the 2nd International Conference on Simulation and Modeling Methodologies, Technologies and Applications* (SIMULTECH-2012), pages 449-452
ISBN: 978-989-8565-20-4

The paper is organized as follow. In section 2 practical considerations are illustrated on how to implement srTCM in OPNET Modeler. In section 3 we will test our implementation. Concluding remarks are given in section 4.

2 IMPLEMENTATION

The configuration of the srTCM is done by assigning three traffic parameters: the Committed Information Rate (CIR), the Committed Burst Size (CBS) and Excess Burst Size (EBS). The way in which the marker should work is presented in (Heinanen and Guerin, 1999a). The behavior is specified in terms of two token buckets: a bucket "C" associated with the CBS value, and a bucket "E" associated with EBS. Both buckets share a common rate CIR.

Initially the token count for each bucket equals the maximum size of that token bucket, meaning that both buckets are full. The token count for bucket C (TC) equals CBS and for bucket E (TE) equals EBS. The token counts TC and TE are updated CIR times per second in the following manner: if TC is less than CBS, TC is incremented with one token, else if TE is less than EBS then TE is incremented with one token. If both token counts are at their maximum values then neither TE nor TC can be incremented. Figure 1 illustrates the srTCM operations.

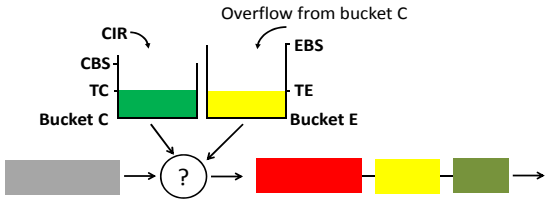

Figure 1: The operations of srTCM.

OPNET implements a policer using the Committed Access Rate (CAR) algorithm (OPNET, 2010). This type of policer enables the device to limit any input or output transmission rate on a specific interface. More information on how the CAR algorithms functions can be found in the Cisco documentation (Cisco, 2012). So as to implement the srTCM algorithm in OPNET we are going to start from the already implemented CAR algorithm.

The only difference that exists between the srTCM specification and our OPNET implementation is that in the latter case the EBS is not specified directly, but must be derived from another parameter, named Extended Burst Size (XBS). Because XBS represents the total volume of the two token buckets, EBS must be equal to the difference between XBS and CBS.

In what follows we present the pseudo code used for implementing srTCM in OPNET Modeler.

Pseudo code used for updating the token bucket:

```
total tokens = total tokens + CIR *
  (simulation time – update time);
update time = simulation time;
if (total tokens > CBS)
 {if (debt > 0)
    {if (total tokens - CBS >= debt)
       {total tokens = CBS;
        Debt = 0;}
     else
  {debt = debt – (total tokens –  CBS);
       total tokens = CBS;}
    }
 else total tokens = CBS;
 }
```

The first part of the pseudo code is in charge of updating the token bucket; this code is already partially implemented in OPNET as part of the CAR algorithm. Figure 2 illustrates the behaviour of the implemented srTCM in OPNET.

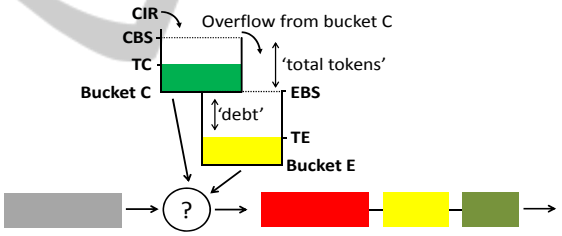

Figure 2: Implementation of srTCM.

The variable total tokens implements simultaneously two functions. First it is used to register the total amount of tokens that are available after a certain amount of time. And secondly, it is used to implement the token bucket C (as is defined in the RFC), where the value of this variable represents in fact what is defined as TC. This variable ranges from the value '0' (when no tokens are available in bucket C) to a value equal to CBS (when the bucket is full).

The debt variable presented in the pseudo-code, represents the free space in the second bucket (bucket E). The value of this variable can be used to retrieve the value of TE, where TE is equal to EBS minus the value of debt. The variable debt ranges from a value equal to EBS (when no tokens are available in bucket E), to a value of '0' (when

bucket E is full of tokens).

Previously we discussed the double functionality of the total tokens variable. This duality is used in the update of the token buckets. We can see in the presented pseudo code that after the update of tokens, an if statement checks if the total amount of tokens is bigger than the CBS. This condition is necessary so as to comply with the original specifications of the marker. In the simple case that the value is less than CBS, no other actions are necessary for updating the token buckets.

On the other hand if the value is bigger than CBS it is checked whether the second bucket is full or not. If the debt is bigger than zero, meaning that the bucket is not full, the excess tokens from the first bucket are put in the second bucket up to the value of the Excess Burst Size. If more tokens than necessary are available, they are just ignored and the token counts for both buckets (total tokens and debt) are put to their maximum value. This being also the case when the two tokens buckets are full meaning that total tokens is bigger than CBS and that debt equals to 0.

Next the pseudo code used for implementing the actual decision making or the actual marking procedure is presented.

Originally it is assumed that the packet is conforming or "green". It is then checked whether the packet size is less than the token count of the first bucket. If this is true the packet size is subtracted from the total amount of tokens. However, if the packet size exceeds the value of the total tokens variable, it is checked if there are enough tokens in the second bucket. This is done by comparing the value of the debt variable plus the packet size against the EBS. If the value is bigger, the packet is marked as violating, or "red", and if the value is less, then the packet is marked as exceeding or "yellow".

Pseudo code used for the marking procedures:

```
if(packet size < total tokens)
 {total tokens = total tokens -
 packet size;}
else
 {
potential debt = debt +
  packet size;
 if(potential debt > EBS)
       {traffic_violate = true;}
 else
       {traffic_exceed = true;
       debt = potential debt;}
 }
```

3 MODEL TESTING

In order to test the presented implementation we constructed a simulation scenario that uses as edge policing the modified CAR algorithm which behaves as a srTCM marker. The network topology used to test our implementation can be seen in figure 3. All the links in the topology are 100BaseT duplex links which operate at 100 Mbps, except for the bottleneck link connecting the two routers in the center of the topology. This bottleneck link is a point to point DS1 link which operates at 1.544 Mbps.

Four pairs of traffic sources and sinks are

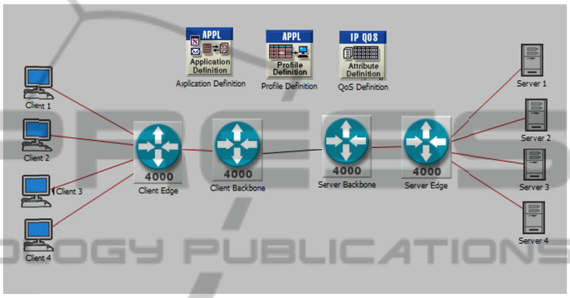

Figure 3: Network Topology.

defined. All the traffic flows from the server (right hand side) to the client (left hand side).

Table 1 presents the associated traffic parameters. All the traffic is defined as belonging to the Assured Forwarding (AF) class 2, having the lowest level of drop probability (AF 21). The srTCM implementation should be able to mark a packet as conforming (keep the AF21 class) if the packet size is smaller than TC. The biggest possible value of TC is CBS, that in this scenario is set to 1 Mbit. The

Table 1: Traffic parameters.

Source	Sink	Applications			
		FTP	Inter request time (seconds)	File size (bytes)	
Server 1	Client 1		exponential (360)	50 000	
Server 2	Client 2		exponential (360)	25 000 000	
Server 3	Client 3		exponential (150)	5 000 000	
Server 4	Client 4	HTTP(2 types of objects/ page)	exponential (1)	object 1	constant (1000) (x1 object /page)
				object 2	uniform (2000, 10000) (x7 objects/ page)

packets should be marked as exceeding (AF 22 class) if the size is bigger than TC, but not bigger than TE. The value associated with the XBS in this case is 50 Mbit. And finally, the packet should be marked as violating (AF23 class) when the packet size exceeds both TC and TE.

Figure 4 presents the statistics collected at the Client Backbone node. We can see in figure 4 that the srTCM implementation marks the traffic flow with three different DSCPs depending of the volume of traffic and on the value of the three parameters (CIR, CBS and EBS).

The traffic of the AF 2 class associated with the lowest level of drop precedence (AF21) is colored green and varies slightly above the value of the defined CIR of 100000 bits. The highest value of around 200000 bits can be noticed in the first seconds of the simulation. This type of behavior is normal, because in the beginning the first token bucket is full and tokens can be "borrowed" from this bucket. But, after some time the bucket becomes empty and since traffic is arriving faster than the refill rate of the tokens, the bucket does not have time to get refilled and the traffic will be marked as "green" (AF21) only up to the value of the defined CIR. However, we can notice that the green traffic is constantly above this CIR. The reason for this is that violating traffic marked as AF23 does not consume any tokens, thus giving the chance to the token bucket to accumulate tokens.

Concerning the second level of drop precedence (AF22) colored yellow, we can see that the tokens associated with this class are all consumed in the first minute of the simulation. Afterwards, because the traffic volume surpasses the CIR, the second bucket does not get a chance to receive any tokens.

This type of behavior is due to the fact that the srTCM recommendation specifies that the second bucket can be refilled only if the first bucket is full.

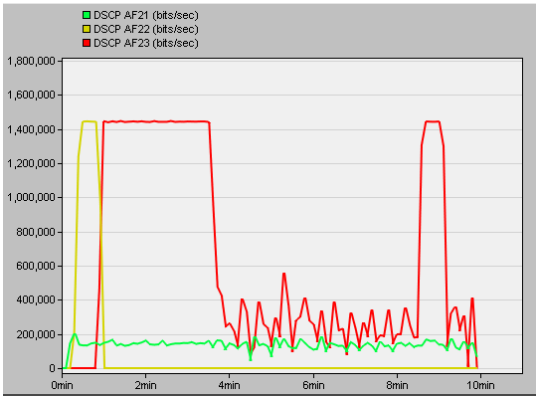

Figure 4: Obtained results.

This kind of problem can be avoided using trTCM instead of srTCM. With trTCM each token bucket has associated a distinct rate which refills the bucket.

The third level of drop precedence AF23 (colored red) is associated will all the traffic which does not receive tokens from any of the token buckets. We can see that this type of marking starts only after all the tokens are consumed.

4 CONCLUSIONS

The current paper presents an implementation of the srTCM in OPNET Modeler. Pseudo code used for this implementation is presented. The implementation itself is tested to demonstrate the accuracy of the implementation. The model testing section of this paper demonstrates that the behavior associated with the modified CAR algorithm is identical to a srTCM behavior.

REFERENCES

Babiarz, J., Chan, K., Baker, F., 2006, Configuration Guidelines for DiffServ Service Classes, RFC 4594.

Blake, S., Black, D., Carlson, E., Davies, E., Weiss, Z., 1998, An Architecture for Differentiated Services, RFC 2475.

Cisco Inc, 2012, Cisco IOS Software Releases 11.1, [online] Available at: < http://www.cisco.com/en/US/docs/ios/11_1/feature/guide/CAR.html >, [Accessed 20 February 2012].

Fang, W., Seddigh, N., Nandy, B., 2000, A Time Sliding Window Three Colour Marker(TSWTCM), RFC 2859.

Heinanen, J., Baker, F., Weiss, W., Wroclawski, J., 1999, Assured Forwarding PHB Group, RFC 2597.

Heinanen, J., Guerin, R., 1999a, A Single Rate Three Color Marker, RFC 2697.

Heinanen, J., Guerin, R., 1999b, A Two Rate Three Color Marker, RFC 2698.

Nichols, K., Blake, S., Baker, F., Black, D., 1998, Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, RFC 2474.

OPNET 2010, OPNET Modeler Documentation Set, Version: 16.0, 10/14/10.

OPNET Inc., 2012, [online] Available at: <http://www.opnet.com> [Accessed 10 February 2012].