USE OF THE NETWORK SIMULATOR NS-2 TOOL IN LECTURES

Petr Berka, Petr Hujka

Department of Telecommunications, Brno University of Technology, Purkynova 118, 612 00 Brno, Czech Republic, phone: +420 5 41149190, utko@feec.vutbr.cz

This article deals with use of Network Simulator ns-2 and Nam tools in networking courses and research at Department of Telecommunication at Brno University of Technology. The OTcl script construction steps are described on the model of computer network implementing Differentiated Services (DiffServ) technology. It provides an overview of ns-2, a short explanation of DiffServ architecture and a brief description of DiffServ implementation in the ns-2 network simulator.

Keywords: Computer networks, Simulator, Protocol, Differentiated Services

1. INTRODUCTION

The Network Simulator – **ns-2** [1] is an object oriented, discrete event driven simulator primarily targeted at networking research. At universities we can use it for teaching students at networking laboratory too, e.g. how computer network design can be done, to protocol, traffic and routing studies, network protocol comparison, etc. The use and work with the ns-2 network simulator are explained by the example of Differentiated Services architecture implementation to ns-2.

Differentiated Services [2] architecture is one of the Quality of Service (QoS) technologies in IP networks. Quality of Service is a general notion of introducing different quality levels to the best-effort Internet. Differentiation of traffic is needed both to guarantee quality requirements for real-time applications as well as to offer different levels of priority inside traffic classes. DiffServ is a set of mechanisms aiming at providing scalable differentiation where quality treatment inside the network is done to aggregates of flows not to individual flows. Quality guarantees can be divided into two service models. In assured service the flows should receive a rate at least equal to the contracted rate, while in relative services the rate of the flows should be proportional to the contracted rate.

In our Department of Telecommunications, the Network Simulator ns-2 is primarily used in research however students familiarize themselves with the ns-2 simulator in networking courses and use it on their bachelor and master theses. The OTcl script construction steps are explained by the implementation of DiffServ architecture to ns-2 simulator.

2. THE NS-2 NETWORK SIMULATOR OVERVIEW

Ns-2 is freely distributed multiplatform open source software. It was developed through the WINT project supported by UC Berkeley, LBL, Xerox PARC, UCB and USC/ISI. Since it is widely used in the research community, a large number of network components are available for ns-2. By reusing these components, the development of complex network architectures can be considerably facilitated.

Among others, ns-2 supports the following technologies:

- Point-to-point connections, LANs, mobile and satellite networking,
- IP, Mobile IP, multicast (DVMRP, PIM, etc.),
- Unicast, multicast and hierarchical routing,
- TCP, UDP, and several experimental transport protocols,
- RTP/RTCP, SRM, QoS (InterServ, DiffServ), QoS routing,
- Applications (Telnet, FTP, WWW-like traffic, etc.), mathematical support, network emulation.

The simulator framework uses a split-language programming approach. The actual core of the simulator is written in C++ to allow for a fast simulation of large scenarios. OTcl, an object oriented version of Tcl language, is used for the control structure and the description of the simulation scenarios. Also the scheduling of events and the dynamic configuration of network components during the simulation is usually done in OTcl. This approach is very flexible but necessity is to have knowledge in OTcl as well as C++. Simulation results can be visualized using the Network Animator – **nam** [1]. Nam can be used to create network topology and simulate various protocols and traffic sources without type Tcl code, only by dragging components by mouse.

3. DIFFERENTIATED SERVICES QOS ARCHITECTURE

Differentiated Services (DiffServ) [2], [3] is a mechanism by which we can offer different levels of network service to the different traffic streams. The foundation of the DiffServ network is that routers within the network handle packets from the different traffic flows by applying different per-hop behaviours (PHB). The PHB to be applied is specified by the DiffServ Code-Point (DSCP) in the IP header of each packet. Advantages of per packet coding are:

- Classification of per packet is done only once (in the access router of the network).
- Forwarding of the packets is based only on the class of packet. There are only a limited number of classes, i.e. different forwarding behaviours.
- DiffServ need neither signalling nor state based reservation of resources inside the network.

DiffServ network uses in general three types of devices: access, core and border router. Access routers classify and assign packets to their particular classes, core routers do forwarding based on the information in the DSCP field. Border routers connect two network providers and do both of the actions of access and core routers. Fig. 1 shows functional mechanisms of access and core routers.

DiffServ maps the service level agreement (SLA) [1], [3] to more detailed service level specification (SLS). A part of SLA, the traffic condition agreement (TCA) is mapped into traffic condition specification (TCS). A simplicity of routers makes DiffServ architecture the most suitable to implement in backbone networks.

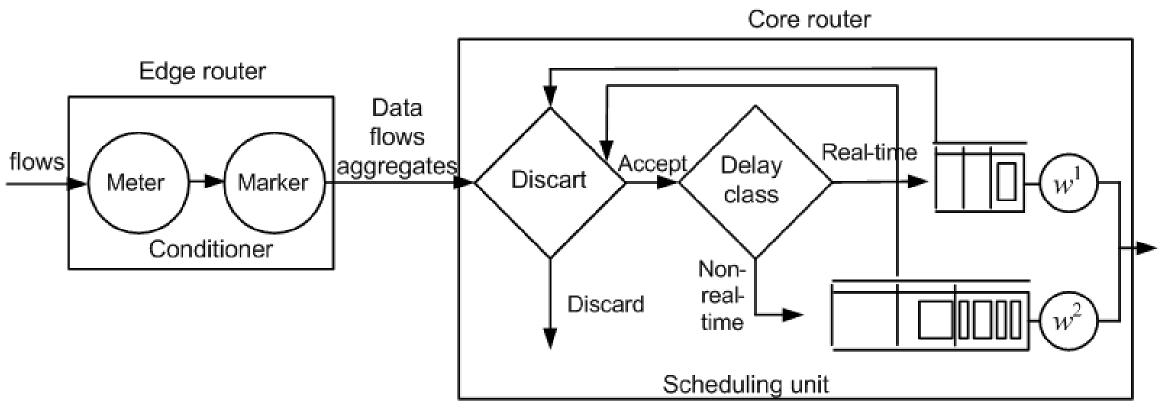

Fig. 1 Functional mechanisms of DiffServ routers

Per-hop behaviour (PHB) is mechanism used for forward the traffic of different service levels with different behaviours. PHB are generally implemented as queue space management in router's output processors. Only two PHBs were adopted as the standards – Expedited Forwarding (EF) [4], [3], and Assured Forwarding (AF) [5], [3]. DiffServ architecture can be implemented in end-to-end delivery model. But, this is not likely to happen, because all the service providers would have to use the same PHBs with the same policies and parameters.

4. EXAMPLE OF DIFFSERV IMPLEMENTATION IN THE NS-2

This example provides a short description of DiffServ architecture implementation to the ns-2 simulator. It explains how the structure of OTcl simulation script has to be written.

4.1 DIFFSERV NETWORK DETAILS

The described DiffServ network consists of one core router (C), two edge routers (E1, E2) and three clients (sources S1, S2 and destination D). The duplex links between S1 – E1, S2 – E1 and E2 – D have 10 Mbps of bandwidth and 5ms of delay. Each node of duplex link uses a FIFO queue, e.i. Droptail in terminology of the ns-2. In the core of the network we have to use simplex links. It is because of DiffServ architecture. Parameters of E1 – C, C – E1 [C – E2 and E2 - C] links are: 10 Mbps, 5 ms [5 Mbps, 5 ms] and dsRED queue. Edge routers have to implement policy functionality – through the policing tables and PHB specified by DSCP. Core routers have to implement only PHB functionality. Sources S1 and S2 generates constant bit rate udp real time traffic, the packet size of traffic from source S1 is 1500 B with rate of 1024 Kbps, from source S2 is 100 B with rate of 128 Kbps. The cbr traffic from S1 starts at 0.5 s and stops at 5 s, cbr traffic from S2 starts at 1 s and stops at 5.5 s. The network topology is as shown in Fig. 2.

4.2 THE ESSENTIAL STEPS OF DIFFSERV SIMULATION

This section shows the constituent steps of the OTcl simulation script construction [1]. It shows only one step of each definition section of the script, i.e. definition of only one link of all, only one router of all usw.

At first we have to define a new simulation process by making a Simulator object instance. Then the nodes must be defined (*node* = S1, S2, D, E1, E2 and C) and the duplex/simplex links between routers. The duplex-links queues are DropTail, simplex-links queues are dsRED/edge or dsRED/core – depends according to the direction.

```
set ns [new Simulator]
set node [$ns node]
$ns duplex-link $S1 $E1 10Mb 5ms DropTail
$ns simplex-link $E1 $C 10Mb 5ms dsRED/edge
```

We have to set up the queues on the corresponding links and their parameters.

```
set dsREDqE2C [[$ns link $E2 $C] queue]
$dsREDqE2C meanPktSize 1500
$dsREDqE2C set numQueues_ 1
$dsREDqE2C setNumPrec 2
$dsREDqE2C addPolicyEntry [$S1 id] [$D id] TokenBucket 10 100000
10
$dsREDqE2C addPolicyEntry [$S2 id] [$D id] TokenBucket 10 300000
40000
$dsREDqE2C addPolicerEntry TokenBucket 10 11
$dsREDqE2C addPHBEntry 10 0 0
$dsREDqE2C addPHBEntry 11 0 1
$dsREDqE2C configQ 0 0 30 50 0.02
$dsREDqE2C configQ 0 1 10 30 0.10
```

Configuration of the queues in the core of network is almost the same as in edge routers except for policy tables. It is sufficient to set up PHB parameters only.

```
set dsREDqCE2 [[$ns link $C $E2] queue]
$dsREDqCE1 meanPktSize 1500
$dsREDqCE1 addPHBEntry 10 0 0
$dsREDqCE1 addPHBEntry 11 0 1
$dsREDqCE1 configQ 0 0 30 50 0.02
$dsREDqCE1 configQ 0 1 10 30 0.10
```

The next commands set up UDP CBR connection between source S1 and destination D.

```
set udp_s1 [new Agent/UDP] – set up the UDP CBR connection
between S1 and D
$ns attach-agent $S1 $udp_s1
```

```
set cbr_s1 [new Application/Traffic/CBR]
$cbr_s1 attach-agent $udp_s1
$cbr_s1 set type_ CBR
$cbr_s1 set packet_size_ $1500
$cbr_s1 rate_ 1024K
$cbr_s1 random_ false
set null0 [new Agent/Null]
$ns attach-agent $D $null0
$ns connect $cbr_s1 $null0
```

The next block of a script creates the scheduler points to schedule events for agents – sending traffic to the destinations, detach agents, finish the simulation, print and animate (Fig. 2) simulation results. The *printStats* command writes out simulation results.

```
$dsREDqE1C printPolicyTable
$dsREDqE1C printPolicerTable
$ns at 0.0 "$recording"
$ns at 0.5 "$cbr_s1 start"
$ns at 1.0 "$cbr_s2 start"
$ns at 1.5 "$qCE2 printStats"
$ns at 3.0 "$qCE2 printStats"
$ns at 4.5 "$qCE2 printStats"
$ns at $5.0 "$cbr_s1 stop"
$ns at $5.5 "$cbr_s2 stop"
$ns at 6.5 "$qE1C printStats"
$ns at 6.5 "$qE2C printStats"
$ns at 6.5 "$qCE1 printStats"
$ns at 6.5 "$qCE2 printStats"
$ns at [expr $testTime + 1.0] "finish"
```

The last command of the script runs the simulation.

```
$ns run
```

We can also include the commands to run and see the simulation in the Network Animator (nam).

```
proc finish {} {
  global ns namfile
  $ns flush-trace
  close $namfile
  exec nam -r 2000.000000us /diffserv.nam &
  exit 0
  }
$ns at 60.000000 "finish"
```

We can compare simulation results between DiffServ and best-effort traffic if we replace dsRED queues with the DropTail queues and run simulation once more.

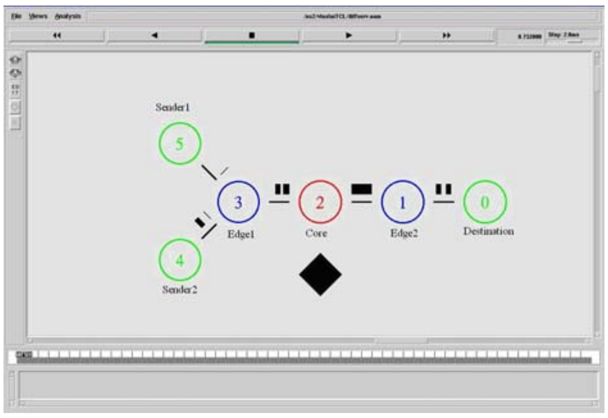

Fig. 2 Output from the network simulator nam

The ns-2 simulator writes simulation results in the table as shown below.

```
Packets Statistics
=======================================
CP  TotPkts   TxPkts   ldrops   edrops
--  -------   ------   ------   ------
All    29994    25023     1507     3464
10    10013     9985       28        0
11    19981    15038     1479     3464
```

5. Conclusion

This paper shortly describes the essentials of The Network Simulator ns-2 tool, it introduces to Differentiated Services architecture of Quality of Service technology and gives an overview how to write OTcl scripts in the ns-2 simulator. The script construction steps are briefly discussed in the example of Differentiated Services architecture implementation.

6. References

[1] Fall, K., Varadhan, K. *The ns manual*. http://www.isi..edu/nsnam/ns/doc/ns_doc.pdf, UC Berkeley, LBL, USC/ISI and Xerox PARC., 2003

[2] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W., *An Architecture for Differentiated Services*. RFC 2475, IETF Networking Group, 12/1998, http://www.ietf.org/rfc/rfc2475.txt.

[3] Luoma, M. *Simulation Studies of Differentiated Services Networks*. Ph.D. Thesis, HUT, Department of Electrical and Communications Engineering, 2000.

[4] Jacobson, V., Nichols, K., Poduri, K. *An Expedited Forwarding PHB*. RFC 2598, IETF Networking Group, 06/1999, http://www.ietf.org/rfc/rfc2598.txt.

[5] Heinanen, J., Baker, F., Weiss, W., Wroclawski, J., *Assured Forwarding PHB Group*. RFC 2597,IETF Networking Group, 06/1999, http://www.ietf.org/rfc/rfc2597.txt.