TKN Telecommunication Networks Group

Technische Universität Berlin
Telecommunication Networks Group

Circuit Switching Module for ns-2

Filip Idzikowski
idzikowski@tkn.tu-berlin.de

Berlin, March 2009

TKN Technical Report TKN-09-003

TKN Technical Reports Series Editor:
Prof. Dr.-Ing. Adam Wolisz

Abstract

This report presents the Circuit Switching Module for ns-2. This module allows establishment of circuits (connections with reserved bandwidth) in the simulated packet-switched network. The Admission Control mechanism at the edges of the circuit decides which packets are allowed to use the circuit. The packets that do not fit into the circuit can be redirected into the packet-switched part of the network, to any outgoing link of the node. Moreover, a Stats Monitor was implemented to collect various packet statistics. The extension was validated using a six node butterfly network with a bidirectional circuit.

Contents

1 Introduction **2**
1.1 Motivation 2
1.2 Metrics 3

2 Simulation model **3**

3 Solution **5**
3.1 Classifier 6
3.2 Queues 6
3.3 Stats Monitor 7

4 Scope of the experiments **8**
4.1 Multipathing 9
4.2 Circuit capacity 9
4.3 Redirections 10

5 Conclusions **12**

A Application Programming Interface **13**
A.1 TCL Syntax 13
A.1.1 CSQueue object commands 13
A.1.2 CSSrcDestFidHashClassifier object commands 14
A.1.3 StatsMonitor object commands 14

References **15**

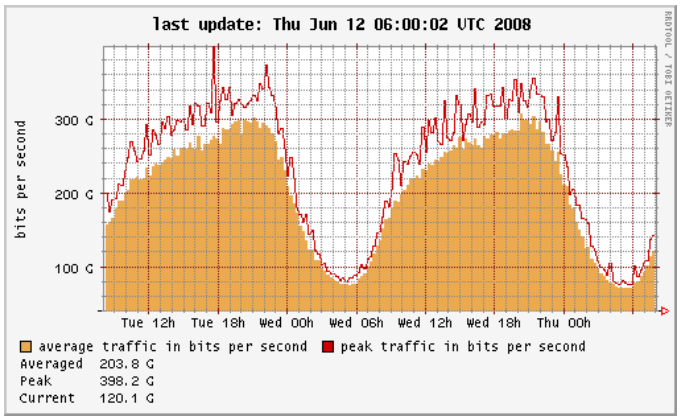

Figure 1: Data rate at DE-CIX (Daily Graph from [3])

1 Introduction

Circuit switching has been present in telecommunication networks since the invention of telephone. Dynamic establishment of circuits may increase the utilisation of transport networks and QoS of Internet applications. In order to investigate these potential benefits using one of the most popular network simulators ns-2 [1], the simulator had to be extended with a Circuit Switching Module. This work describes some implementation details of the Circuit Switching Module, its functionalities, and TCL syntax.

1.1 Motivation

The circuits (connections with reserved banwidth) are to carry IP packets. They are seen as direct links by the IP layer. The data rate of IP traffic is variable in time (see e.g. Fig. 1), so it would be efficient to establish and tear down circuits dynamically, according to the variations of traffic data rate. Admission Control at the edges of circuits makes decisions which packets are allowed to use the circuit. It is a mean of prioritisation (in a special case all packets are allowed to use the circuit). Introduction of shortcut circuits into the packet-switched Internet [2] brings the risk of TCP mechanisms greedily grabbing all the available resources of the circuit. If the packets that do not fit into the circuit (due to lack of its resources) are redirected into the packet switched part of the network, they may be overtaken by subsequent packets transported by the circuit. This may lead to reorderings at the destinations, and therefore to retransmissions, and consequently decrease of goodput as well as increase of jitter at the application level. The Circuit Switching Module for ns-2 has to model the redirections too in order to enable investigation of their influence on the network performance. Moreover, the existing traffic sink models of ns-2 do not contain Flow Monitors that collect statistics like packet delay, jitter etc. (analysis of output trace files demands too much computing resources when simulating

backbone networks), so this is also a subject of simulation model extension.

The simulation model developed in this work may serve to answer the following questions:

- What is the influence of circuit establishment on the performance of the rest of the packet-switched network?
- How should a circuit be dimensioned?
- Between which nodes should a circuit be established?
- How do packet redirections influence network performance?

However the Circuit Switching module can obviously be used for other purposes as well.

1.2 Metrics

In order to be able to measure the improvement or deterioration of the network performance upon the circuit establishment, the following metrics are used:

- Packet transmission delay
- Jitter
- Packet loss rate
- Throughput and goodput
- Network and link utilisation

They are collected by the implemented Stats Monitors (see section 3.3). Various statistics can also be obtained by analysis of the output trace file of ns-2.

2 Simulation model

The base of the simulation model is inherited from ns-2 [1]. The Circuit Switching Module as extension to ns-2 is presented in this section. It possesses the following functionalities:

- Each node can assign packets to a packet-switched part of the network (PSPN, lower priority) and circuit-switched part of the network (CSPN, higher priority).
- A circuit can be established between any two nodes in the network.
- Multiple circuits can be established between a node pair.

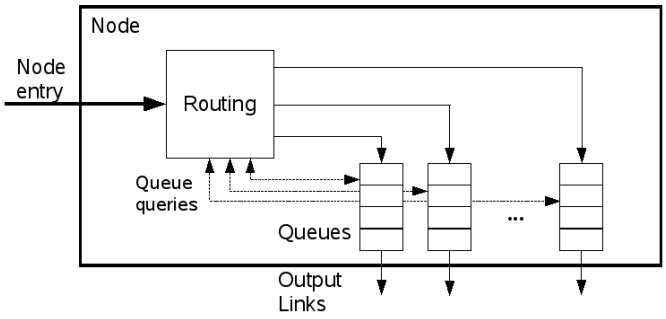

Figure 2: Simplified node model without traffic sources and sinks

- The capacity reserved for the circuit can take any positive value smaller equal the capacity available at the links the circuit traverses.
- The circuit can take arbitrary, user-defined routes.
- Each circuit possesses a buffer at its edge. The buffer acts as a traffic shaper. It stores the packets in case a burst of packets temporarily exceeds the capacity of the circuit, and decreases the packet losses at the edge of the circuit.

The considered system consists of a group of network nodes interconnected by links. A simplified node model is shown in Figure 2. Packets arrive at the node entry and the routing decision is made according to their source, destination and flow id as well as the state of the queues at the node's outputs. The queues are served by the output links with their data rates.

Figure 3 shows the structure of a single queue from Figure 2. It stores packets that are to be sent to one output link. In order to guarantee the capacity for a circuit, not only appropriate bandwidth of links needs to be exclusively reserved for the circuit, but also the buffer capacity needs to be exclusively reserved for it. Therefore the buffer at the input to a link is divided into up to d separate buffers, one of which is dedicated to PSPN, and the rest to the circuits traversing given node. A packet upon arrival is enqueued in the appropriate buffer based on the flow id of the incoming packet and its status (whether it is a redirected packet or not - see section 3.1). If the incoming packet arrives to a full PSPN buffer, the packet is dropped. Packets at the head of each subqueue are served at the rates assigned to corresponding circuits during their establishment.

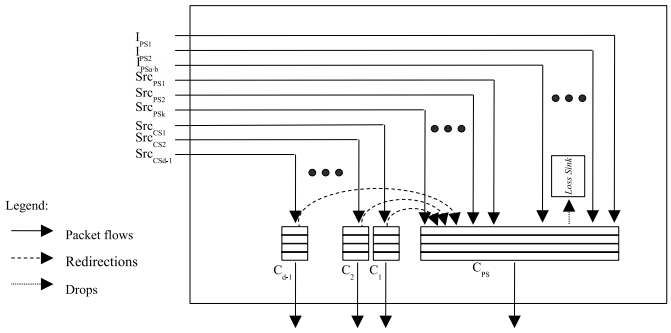

Figure 3: Structure of a queue at the entry of a link

3 Solution

ns-2 is an event based simulator. The following types of events can be distinguished: generation of a packet, arrival of a packet at a node, completion of packet transmission. It would be optimal to model circuits as separate links between the circuit source and destination, and keep the just mentioned types of events unchanged. However, since a link in ns-2 is determined by the nodes it spans, it is impossible to establish parallel links, which could emulate separate circuits (or wavelength channels of a fibre). The bandwidth reservation is achieved by Time Division Multiplexing (TDM). Time is divided into frames which consists of time slots. The time slots correspond to circuits. In order to reduce the computational complexity of the simulation model, the time slots can have unequal durations, so that the desired bandwidth can be reserved for each circuit. Introduction of TDM demands different behaviour of the network upon occurrence of the events (especially regarding the beginning of service).

This behaviour is enclosed in three main blocks of the Circuit Switching Module:

- a classifier (within a new Circuit Switching Routing Module) which classifies (and according to ns-2 implementation also forwards) packets into the appropriate queue of the queueing system or directly to the receiver
- a queueing system that stores packets corresponding to the output link they head for
- a statistics monitor which receives packets at their destination and updates various statistics

These blocks are discussed in the following sections. Please note that the Circuit Switching Module was implemented and tested with the version 2.29 of the network simulator. We also validated the compatibility of the complete module with version 2.33 of ns-2.

3.1 Classifier

The Classifier is one of the most important objects in ns-2. It classifies packets according to the precomputed routing tables, and forwards them to the appropriate outputs of a node (including the traffic sinks). We use the *HashClassifier* as the base class for our classifier (*CSSrcDestFidHashClassifier*). It can make routing decisions based on the source, destination and flow id of a packet.

Although the mapping between the incoming packet's flow id and the resources it is going to use is made in the queueing system of an output link itself, it is the classifier that realizes redirections. Redirection of packets is nothing else than a routing decision dependent on the fill level of the queue the packet is going to be stored in (at the edge of an output link of a node). This is shown in Fig. 2. An access of the classifier to the queue had to be implemented, as the classifiers available in ns-2 make the routing decision before checking the fill level of the queue. Redirected packets get new packet ids, as indication to classifiers about their ”redirected” status. A summary of properties of the classifier is as follows:

- a possibility to check the fill level of the queue a packet is going to be stored in
- forwarding packets from node's entry to appropriate output link or sink, based on the src, dst and fid of the packet, and fill level of the corresponding queue
- a possibility to redirect packets to the arbitrarily prespecified output links in the packet-switched part of the network
- turning the redirections on and off separately for each node
- marking of redirected packets with a unique flow id using appropriate queue

3.2 Queues

The queueing system at an output link of a node is quite complex, as indicated in section 2. A queue consisting of several subqueues is needed. Therefore a *CSQueue* class has been implemented, which contains objects of class *SlotQueue*. Each *SlotQueue* is responsible for a separate TDM time slot. A weight is associated with each time slot to indicate the bandwidth assigned to it. Special scheduling system had to be designed to take care of the bandwidth sharing. Current time slot needs to be determined at arrival of each packet at a node. This determines whether the packet can be transmitted straight away to the next node (under the condition that no packet of the same type is currently being transmitted). Moreover, the remaining resources (up to the end of the time slot) need to be sufficient to complete the transmission of a packet.

The bandwidth is shared in a TDM manner in this implementation, however it can mimic WDM too via sufficiently small time frames. The packets which do not

fit into a time frame are not fragmented, but need to wait for the following time frame. Mapping between the flow ids and time slots is stored in a Flow Behaviour Table (FBT). Entries of FBT need to be defined prior to the start of the simulation. The *slotQueues* are dropTail FIFO queues, however the Circuit Switching Module allows implementation of other queueing schemes.

3.3 Stats Monitor

In order to collect network statistics a statistic monitor class *StatsMonitor* was implemented. This is an agent located at the destination node of a packet flow. It calculates various statistics up-to-date with the running simulation. Class *AppStatsMonitor* is an extension of the class *StatsMonitor*, which can calculate also the goodput of a flow (important for connection-oriented protocols). It is integrated into the existing base *TcpSink* class in ns-2. The following statistics can be collected using method of batch means and the *(App)StatsMonitors*:

- Over the whole network:
  - amount of received packets
  - average transmission time per packet and the corresponding confidence interval
  - average jitter and the corresponding confidence interval
  - average throughput per flow and the corresponding confidence interval
  - average goodput per flow and the corresponding confidence interval (AppStatsMonitor only)
- For each traffic flow:
  - amount of received packets
  - average transmission time per packet and the corresponding confidence interval
  - average jitter and the corresponding confidence interval
  - average throughput of the flow and the corresponding confidence interval
  - average goodput of the flow and the corresponding confidence interval (AppStatsMonitor only)

The statistics can be written out to text files at each moment of the simulation. Moreover, by postprocessing of the output simulation trace file, the following information can be obtained using the developed software:

- For the whole network:
  - amount of dropped packets

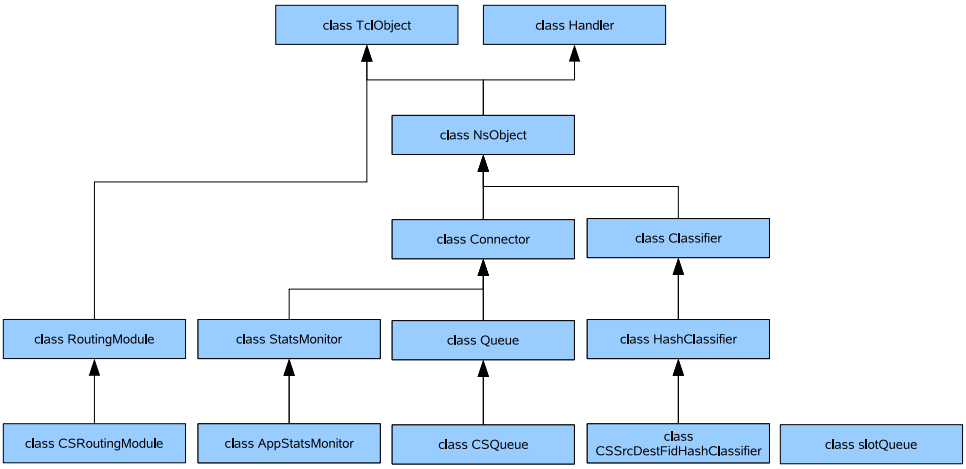

Figure 4: Hierarchy of the implemented classes

  - network utilisation

- For each node:
  - amount of dropped packets
- For each link:
  - utilisation
- For each flow:
  - Ratio of the number of redirected packets to all packets in a flow

Detailed information about the amount of received, transmitted, dropped and redirected packets at each node is available as well.

Figure 4 shows the dependency of the implemented C++ classes from their ancestors.

4 Scope of the experiments

We performed three experiments in order to validate the functionality of our simulation model. The first one checks the possibility to chose an arbitrary path through the network to transport packets. In the second one (which is actually a set of experiments) we analyse the influence of circuits on the performance of the packet switched network, present some results collected by the Statistics Monitors. The third experiment serves for validation of redirections of packets that do not fit into a circuit. All of the expermients use a butterfly network (Fig. 5) with bidirectional STM-64 links (9953.28 Mbps).

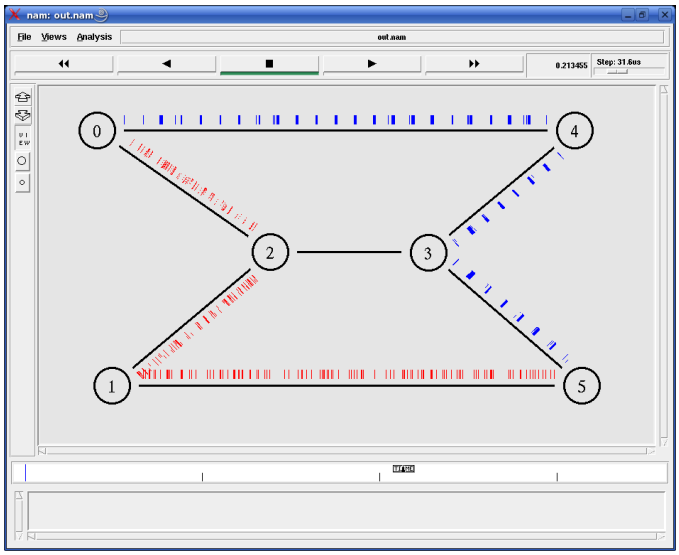

Figure 5: Two flows with the same source (0) and destination (5) nodes, but different flow ids taking different paths - nam graph

4.1 Multipathing

The Circuit Switching Module must allow the simulator user choose a path the circuit is going to take. It does not have to be the shortest path. Moreover, the flows within the same source and destination addresses, but different flow ids may need to be routed via different paths. In order to achieve it in ns-2, the new routing module with *CSSrcDestFidHashClassifier* had to be used. We performed a simple experiment to validate the freedom of the path selection for packets and circuits. Just two traffic sources (located at node 0) generate traffic (both destined to node 5, see Fig. 5). The flows have different flow ids though. By appropriate settings of classifiers at all nodes, as well as the flow behaviour entries at the queues, one flow takes the route via nodes 4 and 3 (via circuit), and the other via nodes 2 and 1 (hop by hop).

4.2 Circuit capacity

A circuit, as a connection with reserved capacity, has to be dimensioned. The capacity of a circuit is determined by duration of its slot within a time frame. We validate the circuit dimensioning using the scenario shown in Fig. 6. There is a bidirectional end-to-end circuit established between nodes 0 and 5, which traverses nodes 4 and 3. The packets between these nodes hop via nodes 2 and 1 in a purely packet switched network (with shortest path routing where all links have the same link weight), which serves as a reference scenario for us. We employ one traffic generator for each source-destination pair (6 x (6-1 = 30) generators altogether). Traffic is generated according to Poisson Model with constant packet size of 1500

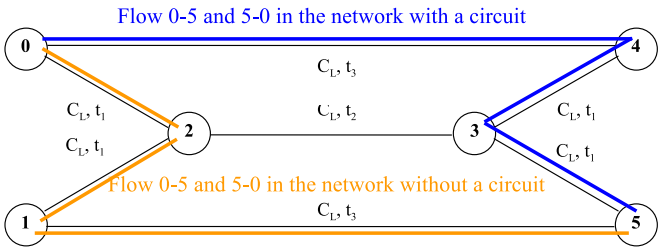

Figure 6: Circuit capacity scenario

B, and average data rate 2.0 Gbps. The circuit capacity is varied between 15 and 60 percent of the link capacity, and the StatsMonitors collect the statistics mentioned in Section 3.3. The results are stored in text files. Some of the results are graphically shown in Figures 7 - 10 (circuit size is shown on the OX axis). The horizontal line shows the reference values for a network without any circuits (purely packet switched).

Introduction of the circuit brings benefits due to better load balance (unloading of the links 0-2 and 2-1). Moreover, if the circuit is dimensioned between 21 and 40 percent of the link capacity, the benefits are maximised for all considered metrics. The plots confirm the intuition that the transmission delay and packet drops increase, and the network utilisation and throughput decrease if the circuit has too little capacity, as well if it is overdimensioned (lacking resources for the PSPN).

4.3 Redirections

Implementation of the redirections and monitoring of TCP flows was validated with the same scenario as the one used in the previous section (Fig. 6). The circuit capacity in this experiment is kept fixed equal to 20 percent of the STM-64 stream. Figures 11 and 12 show the ratio of the amount of redirected packets to the amount of all packets in the flow in dependence of time for load equal to 2.1 and 2.4 Gbps per traffic generator. They include the transient phase of the simulation. For the first case (2.1 Gbps) the redirected packets traverse hop-by-hop the path 0-2-1-5 using the resources of PSPN, and for the second case (2.4 Gbps) they take the same path as the circuit (0-4-3-5). The redirections show the same behaviour at both edges of the bidirectional circuit. Since circuit capacity is close to 2 Gbps, the percentage of redirected packets shown in the Figures increases up to the expected values - 0.052 and 0.170 respectively.

In order to validate the AppStatsMonitor embedded into the TCP Sink, the Poisson Traffic Sources were replaced with TCP Agents, which perform FTP file transfers. The Congestion Control mechanism function well, and the appropriate statistics (including network and flow goodput and their confidence intervals) are collected.

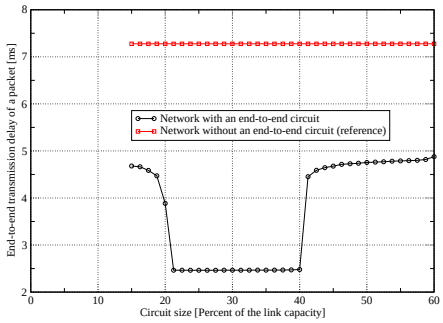

Figure 7: Average packet transmission time in the network

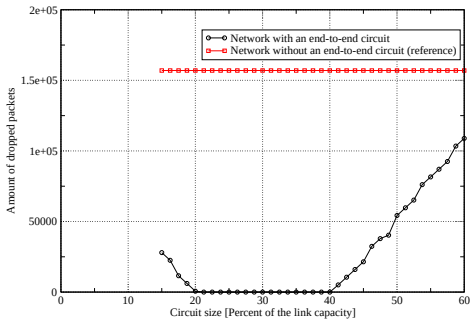

Figure 8: Amount of dropped packet in the network

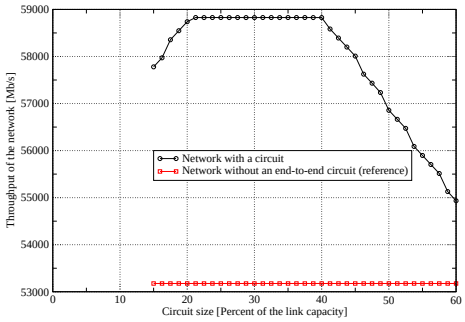

Figure 9: Network throughput

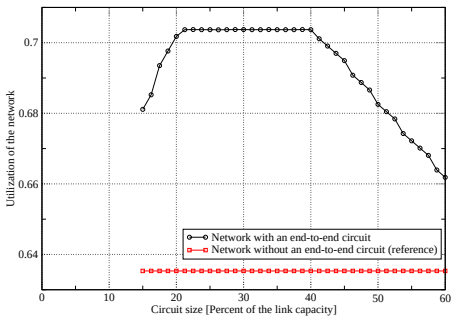

Figure 10: Network utilisation

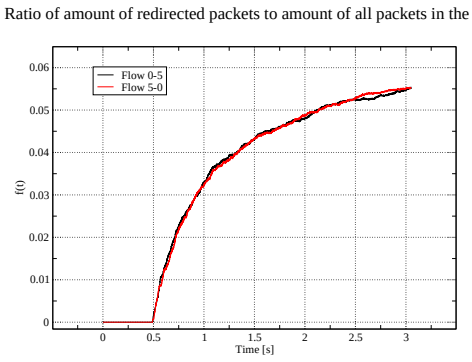

Figure 11: Ratio of amount of redirected packets to the amount of all packets in the flow in dependence of time for the scenario from Fig. 6 and load equal to 2.1 Gbps per traffic source

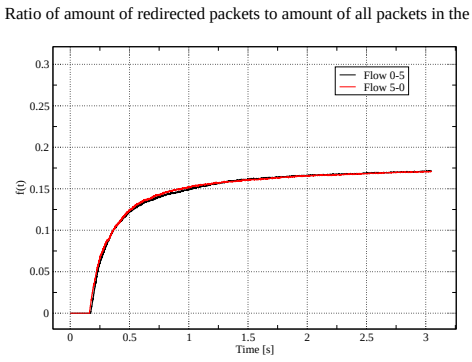

Figure 12: Ratio of amount of redirected packets to the amount of all packets in the flow in dependence of time for the scenario from Fig. 6 and load equal to 2.4 Gbps per traffic source

5 Conclusions

The network simulator was successfully extended and can collect various network, flow, link and node statistics for TCP and UDP traffic. The packets that do not fit into a circuit due to lack of its resources can be redirected to any of the output links of the node. However, variability of data rates the TCP sources transmit traffic with, increases the amount of packets that need to be simulated in order to get sufficient confidence intervals of simulation results. Another issue is that due to the limited TCP window size, multiple TCP connections need to be considered in order to better utilise optical circuits. However, numerous TCP connections may lead to high memory consumption of ns-2. Window-scaling and High-speed TCP are considered here.

It is worth mentioning that the developed module can be integrated with ns-2 without changing or disabling any of the functionalities of the original simulator. It was validated with versions 2.29 and 2.33 of ns-2.

A Application Programming Interface

A.1 TCL Syntax

A.1.1 CSQueue object commands

- *$csqueue setSlotQLimit <numSlot> <queueLimit>* - set the limit of the underlying time slot queue (slotQueue) number *numSlot* to *queueLimit* packets
- *$csqueue addFBEntry <fid> <slot>* - add a Flow Behaviour Table entry mapping the flow with flow id *fid* to the time slot *slot*
- *$csqueue getStat <stat> <slotQueue>* - get a desired statistic (drops, packets, redirections) of the desired subqueue number *slotQueue* and print the result
- *$csqueue getSlotQLength <slotQueue>* - get length of the desired subqueue number *slotQueue* and print the result
- *$csqueue printStats* - print the statistics for this queue (CSQueue)
- *$csqueue setFraming* - set the TDM framing according to the amount of slots and their weights
- *$csqueue setQId <qId>* - set the *qId* for this queue (CSQueue)
- *$csqueue printFBTable* - print the Flow Behaviour Table (FBT)
- *$csqueue addSlotWeight <slot> <weight>* - set the weight of the slot *slot* to *weight* (in bits transmitted in a time frame)
- *$csqueue setQMode <behaviour> <slotQueue>* - set the behavior of the queue *slotQueue* (by default all underlaying queues (slotQueues)) to *behaviour* (so far dropTail is available)
- *$csqueue setLabelBound <labelBound>* - set a static labelBound (for all slotQueues) to *labelBound* (amount of possible flow ids of not redirected flows)
- *$csqueue addFlowInCir <fid> <slotQueue>* - add a flow with flow id *fid* to the flows transported by a circuit in a single slotQueue number *slotQueue*
- *$csqueue setRedirectionsOnOff <value>* - turn on or off (*value*) redirections at a slotQueue number *slotQueue*
- *$csqueue setDefaultRedirectSlot <defaultSlot>* - set a default slotQueue of this CSQueue to *defaultSlot*, where the packets can be redirected to

A.1.2 CSSrcDestFidHashClassifier object commands

- *$cssrcdestfidhashclassifier set-hash <buck> <src> <dst> <fid> <slot>* - inserts a new entry into the hash table within the classifier. The *buck* argument specifies the hash table bucket number to use for the insertion of this entry. When the bucket number is not known, *buck* may be specified as auto. The *src*, *dst* and *fid* arguments specify the IP source, destination, and flow IDs to be matched for flow classification [1]. The slot argument indicates the node to which the packet characterised by *src*, *dst* and *fid* is to be forwarded to.
- *$cssrcdestfidhashclassifier del-hash <src> <dst> <fid>* - removes an entry from the hash table. The entry is characterised by source (*src*), destination (*dst*), and flow ID (*fid*)
- *$cssrcdestfidhashclassifier setRedirectionsOnOff <value>* - turns on or off (*value*) the redirections in the CSSrcDestFidHashClassifier

A.1.3 StatsMonitor object commands

- *$statsmonitor clear* - clears the variable indicating the expected packet to arrive
- *$statsmonitor printStats* - prints the collected statistics
- *$statsmonitor enableStat* - enables collection of statistics by the StatsMonitor
- *$statsmonitor calculatePktBatch* - calculates statistics of a batch of data dependent on the amount of packets in the network
- *$statsmonitor calculateTimeBatch* - calculates statistics of a batch of data dependent on the amount of simulated time (including the goodput for AppStatsMonitor)
- *$statsmonitor resetPktBatch* - resets the variables responsible for calculating statistics of a batch of data dependent on the amount of packets in the network
- *$statsmonitor resetTimeBatch* - resets the variables responsible for calculating statistics of a batch of data dependent on the amount of simulated time
- *$statsmonitor summarizeTimeBatch* - summarises a time batch (once the time of a batch has been reached) for both network and single sinks, and resets the appropriate data to prepare the data to measure the next batch
- *$statsmonitor printNetworkStats <fileName>* - print statistics of all flows in the network to file of name *fileName*

- *$statsmonitor printNetworkStatsBatch <fileName>* - print batch statistics of all flows in the network to file of name *fileName*
- *$statsmonitor setNetworkPktsToSimulateBatch_ <pkts>* - set the amount of packets to be simulated in a single batch to *pkts*
- *$statsmonitor setBatchTime_ <time>* - set the amount of time to be simulated in a single batch to *time*
- *$statsmonitor setID <id>* - set the statsMonitor id to *id*
- *$statsmonitor setMonitorL4Protocol <prot>* - set the layer 4 protocol of the monitored flow to <prot> (udp and tcp available)
- *$statsmonitor setSteadyStateTime time* - set the time where the simulation steady state is assumed to be reached to <time>
- *$statsmonitor printStats <src> <dst> <fileName>* - print statistics for the flow of packets between *src* and *dst* into the file *fileName*
- *$statsmonitor printStatsBatch <src> <dst> <fileName>* - print batch statistics for the flow of packets between *src* and *dst* into the file *fileName*

References

[1] K. Fall, K. Varadhan, and et. al. *The ns manual*. The VINT Project, http://www.isi.edu/nsnam/, February 2008.

[2] A. Feldmann, J. Rexford, and R. Caceres. Efficient Policies for Carrying Web traffic Over Flow-Switched Networks. *IEEE/ACM Transactions on Networking*, 6(6):673–685, 1998.

[3] German Internet Exchange DE-CIX, http://www.de-cix.net/content/network/Traffic-Statistics.html. *DE-CIX Traffic Statistics*, June 2008.