Часть I
Имитационное моделирование в NS-2

I.1. Теоретические сведения

I.1.1. Общее описание NS-2

В данном разделе использованы материалы работ [1; 3; 4].

Network Simulator (NS-2) — один из программных симуляторов моделирования процессов в компьютерных сетях. NS-2 позволяет описать топологию сети, конфигурацию источников и приёмников трафика, параметры соединений (полосу пропускания, задержку, вероятность потерь пакетов и т.д.) и множество других параметров моделируемой системы. Данные о динамике трафика, состоянии соединений и объектов сети, а также информация о работе протоколов фиксируются в генерируемом trace-файле.

NS-2 является объектно-ориентированным программным обеспечением. Его ядро реализовано на языке C++. В качестве интерпретатора используется язык скриптов (сценариев) OTcl (Object oriented Tool Command Language). NS-2 полностью поддерживает иерархию классов C++ и подобную иерархию классов интерпретатора OTcl. Обе иерархии обладают идентичной структурой, т.е. существует однозначное соответствие между классом одной иерархии и таким же классом другой. Объединение для совместного функционирования C++ и OTcl производится при помощи TclCl (Classes Tcl). В случае, если необходимо реализовать какую-либо специфическую функцию, не реализованную в NS-2 на уровне ядра, для этого используется код на C++.

Процесс создания модели сети для NS-2 состоит из нескольких этапов:

1) создание нового объекта класса Simulator, в котором содержатся методы, необходимые для дальнейшего описания модели (например, методы `new` и `delete` используются для создания и уничтожения объектов соответственно);
2) описание топологии моделируемой сети с помощью трёх основных функциональных блоков: узлов (nodes), соединений (links) и агентов (agents);
3) задание различных действий, характеризующих работу сети.

Для создания узла используется метод `node`. При этом каждому узлу автоматически присваивается уникальный адрес. Для построения однонаправленных и двунаправленных линий соединения узлов используют методы `simplex-link` и `duplex-link` соответственно.

Важным объектом NS-2 являются агенты, которые могут рассматриваться как процессы и/или как транспортные единицы, работающие на узлах моделируемой сети. Агенты могут выступать в качестве источников трафика или приёмников, а также как динамические маршрутизирующие и протокольные модули. Агенты создаются с помощью методов общего класса `Agent` и являются объектами его подкласса, т.е. `Agent/type`, где `type` определяет тип конкретного объекта. Например, TCP-агент может быть создан с помощью команды:

```
set tcp [ new Agent/TCP ]
```

Для закрепления агента за конкретным узлом используется метод `attach-agent`. Каждому агенту присваивается уникальный адрес порта для заданного узла (аналогично портам tcp и udp). Чтобы за конкретным агентом закрепить источник, используют методы `attach-source` и `attach-traffic`. Например, можно прикрепить ftp или telnet источники к TCP-агенту. Есть агенты, которые генерируют свои собственные данные, например, CBR-агент (Constant Bit-Rate) — источник трафика с постоянной интенсивностью.

Действия разных агентов могут быть назначены планировщиком событий (Event Scheduler) в определённые моменты времени (также в определённые моменты времени могут быть задействованы или отключены те или иные источники данных, запись

статистики, разрыв, либо восстановление соединений, реконфигурация топологии и т.д.). Для этого может использоваться метод `at`. Моделирование начинается при помощи метода `run`.

В качестве дополнения к NS-2 часто используют средство визуализации nam (network animator) для графического отображения свойств моделируемой системы и проходящего через неё трафика и пакет Xgraph для графического представления результатов моделирования.

Запуск сценария NS-2 осуществляется в командной строке с помощью команды:

```
ns [tclscript]
```

Здесь [tclscript] — имя файла скрипта Tcl, который определяет сценарий моделирования (т.е. топологию и различные события).

Nam можно запустить с помощью команды

```
nam [nam-file]
```

Здесь [nam-file] — имя nam trace-файла, сгенерированного с помощью ns.

I.1.2. Список некоторых команд NS-2

Замечание. При копировании текстов листингов не забывайте убирать знак переноса строки «\» и перенабирать кавычки.

Объекты типа NODE

- `$node id` — возвращает идентификатор узла;
- `$node neighbors` — возвращает список соседних узлов;
- `$node attach agent` — прикрепляет агент типа `agent` к узлу;
- `$node detach agent` — отменяет прикрепление агента типа `agent` к узлу;
- `$node agent port` — возвращает ссылку на агента, прикреплённого к порту `port` на данном узле, или пустую строку, если порт не используется.
- `$node reset port` — отменяет прикрепление всех агентов на данном узле и реинициализирует все переменные, связанные с агентами данного узла.
- `$node join-group agent group` — добавляет объект, определяемый объектной ссылкой `agent` для многопользовательской группы с адресом `group`. Это также приводит к выделению соответствующего многопользовательского трафика протоколом групповой работы для обеспечения работы агента `agent`.
- `$node allocaddr` — возвращает адреса в многопользовательской группе в возрастающем порядке для каждого соединения, начиная с 0x8000 и заканчивая 0xFFFF.

DELAYLINK объекты

Объекты данного типа определяют количество времени, требующегося пакетам для прохождения соединения. Время определяется соотношением

```
size/bw + delay,
```

где `size` — размер пакета, `bw` — ширина полосы соединения, `delay` — задержка распространения соединения.

Конфигурационные параметры:

- `bandwidth_` — ширина полосы передачи соединения в бит/сек;
- `delay_` — задержка распространения в сек.

Методы динамики сети

– `$ns rtmodel model model-params node1 [node2]` — данная команда восстанавливает / разрывает соединение между узлами `node1` и `node2` в соответствии с моделью `model`;
`model-params` содержит все параметры, необходимые для определения модели, и должны быть указаны в виде списка, т.е. параметры должны быть заключены в фигурные скобки;
`model` может иметь одно из следующих значений: `Deterministic`, `Exponential`, `Manual` или `Trace`.
Команда возвращает ссылку на объект модели в соответсвии с определением `model`.
В модели `Deterministic` `model-params` имеют вид:

```
[start-time] up-interval down-interval [finish-time]
```

Начиная с момента `start-time`, соединение восстанавливается при `up-interval` и разрывается при `down-interval` до `finish-time`. Значения по умолчанию для `start-time` — 0, 5 с, `up-interval` — 2, 0 с и `down-interval` — 1, 0 с. Значение по умолчанию `finish-time` — время окончания моделирования.
При использовании модели типа `Exponential` параметры записываются в виде: `up-interval down-interval`. Времена восстановления и разрыва соединения выбираются из экспоненциального распределения со средними `up-interval` и `down-interval` соответственно. Значения по умолчанию для средних `up-interval` и `down-interval` — 10, 0 с и 1, 0 с.
Если используется модель типа `Manual`, в качестве `model-params` применяется `at op`, где `at` определяет время, когда операция `op` должна произойти. Значение `op` может быть `up` или `down`. Альтернативой данной модели может быть метод `rtmodel-at`.
Последний тип `Trace` используется в случае, когда динамика узлов / соединений читается из trace-файла. Аргумент `model-params` представляет собой ссылку на trace-файл, в котором находится информация о динамике.
– `$ns rtmodel-delete model-handle` — уничтожает модель, определённую `model-handle`;
– `$ns rtmodel-at at op node1 [node2]` — используется для указания времени восстановления / разрыва соединения между узлами `node1` и `node2`. Если определён только один узел `node1`, то команда будет применена ко всем соединениям, связанным с данным узлом; `at` определяет время выполнения операции `op`, которая может быть `up` или `down` по отношению к определённому соединению.

Объекты типа QUEUE

Объекты типа очередь — основной класс объектов управления пакетами при их движении по моделируемой топологии.

Конфигурационные параметры:
– `limit_` — размер очереди в пакетах;
– `blocked_` — по умолчанию установлено в false и имеет значение true, если очередь блокирована (не способна посылать пакеты соседнему узлу по соединению);
– `unblock_on_resume_` — по умолчанию установлено в true, показывает, что очередь автоматически разблокируется после передачи последнего полученного пакета.

Подклассы объектов типа Queue

Объекты типа `Drop-Tail` используют простейший алгоритм обработки типа FIFO. Для данного подкласса не определены никакие методы, конфигурационные параметры или переменные состояния.

Объекты типа `FQ` используют алгоритм обработки типа *Fair Queuing (алгоритм организации равноправных очередей)*. Конфигурационные параметры:
- `secsPerByte_`

Объекты типа `SFQ` используют алгоритм обработки типа *Stochastic Fair queuing (алгоритм стохастического справедливого обслуживания)*. Конфигурационные параметры:
- `maxqueue_`
- `buckets_`

Объекты типа `DRR` используют *Deficit Round Robin Scheduling* (алгоритм циклического обслуживания с возможностью обработки сверх нормы некоторого количества пакетов (байт) очереди с последующим уменьшением на эту величину (deficit) при реализации следующего цикла обслуживания). Конфигурационные параметры:
- `buckets_` — показывает общее число областей памяти, используемых для смешивания каждого потока;
- `blimit_` — показывает размер используемого буфера в байтах;
- `quantum_` — показывает (в байтах), как много каждый поток может послать за отведённое ему время;
- `mask_` — когда установлено в 1, означает, что отдельный поток состоит из пакетов, имеющих одинаковые идентификаторы узла (и возможно различные идентификаторы порта), в противном случае поток состоит из пакетов с одинаковыми идентификаторами порта и узла.

Объекты типа `RED` используют алгоритм *Random Early-Detection (алгоритм случайного раннего обнаружения)*. Объект может быть сконфигурирован как для отбрасывания, так и для пометки для отбрасывания пакетов. Конфигурационные параметры:
- `bytes_` — установка в true означает byte-mode RED, где размер прибывающих пакетов влияет на вероятность отбрасывания («отметку») пакетов;
- `queue-in-bytes_` — установка в true показывает, что среднее измерение очереди производится в байтах, а не в пакетах. Установка этой опции также приводит к автоматическому масштабированию `thresh_` и `maxthresh_` с помощью `mean_pktsize_`.
- `thresh_` – минимальная граница для среднего размера очереди в пакетах;
- `maxthresh_` — максимальная граница для среднего размера очереди в пакетах;
- `mean_pktsize_` — оценка среднего размера пакета в байтах. Используется для обновления вычисленного среднего размера очереди после периода ожидания;
- `q_weight_` — вес очереди, используется для вычисления среднего размера очереди;
- `wait_` — при установке в true устанавливается интервал между отброшенными пакетами;
- `linterm_` — как средний размер очереди варьируется между `thresh_` и `maxthresh_`, так и вероятность отбрасывания пакета варьируется между 0 и 1/`linterm_`;
- `setbit_` — при установке в true пакеты не отбрасываются, а помечаются на удаление (в пакетах устанавливается бит, характеризующий перегрузку);
- `drop-tail_` – при установке в true вместо механизма randomdrop используется механизм drop-tail в случае переполнения очереди или когда средний размер очереди превосходит `maxthresh_`.

Объекты типа CBQ (Class-Based Queue) используют методы обработки в зависимости от классов трафика:
- $cbq insert $class — добавляет класс трафика class в структуру распределённых соединений, соответствующую объекту соединения cbq;
- $cbq bind $cbqclass $id1 [$id2] — устанавливает соответствие между пакетами с идентификатором потока $id1 (или диапазона $id1 - $id2) и классом трафика $cbqclass.
- $cbq algorithm $alg — определяет внутренний алгоритм CBQ; $alg может быть установлено в одно из значений: ancestor-only, top-level или formal.

Объекты типа CBQ/WRR используют взвешенное циклическое управление (Round-Robin Scheduling) потоками между классами одного приоритетного уровня. Конфигурационные параметры:
- maxpkt_ — максимальный размер пакета в байтах. Значение этого параметра используется только CBQ/WRR объектами для вычисления максимального выделения полосы для взвешенного циклического управления (Round-Robin Scheduling).

Объекты типа QUEUEMONITOR

Объекты типа QUEUEMONITOR используются для мониторинга получаемых, отправляемых и отбрасываемых пакетов и байтов. Они также поддерживают вычисление статистики (среднего размера очереди и т.д.):
- $queuemonitor — сбрасывает все описываемые далее счетчики (прибывших, отправленных и отброшенных байт) в ноль, а также значения интеграторов и элементов задержки, если это определено;
- $queuemonitor set-delay-samples delaySamp_ — устанавливает Samples объект delaySamp_ для записи статистики о задержках очереди; delaySamp_ — управляющая переменная для объекта Samples, т.е. объект Samples должен быть уже создан;
- $queuemonitor get-bytes-integrator — возвращает состояние объекта типа Integrator, который может быть использован для вычисления интегрального значения размера очереди в байтах;
- $queuemonitor get-pkts-integrator — возвращает состояние объекта типа Integrator, который может быть использован для вычисления интегрального значения размера очереди в пакетах;
- $queuemonitor get-delay-samples — возвращает состояние объекта типа Samples delaySamp_ для записи статистики о задержках очереди.

Переменные состояния:
- size_ — текущий размер очереди в байтах;
- pkts_ — текущий размер очереди в пакетах;
- parrivals_ — общее число полученных пакетов;
- barrivals_ — общее число полученных байт;
- pdepartures_ — общее число отправленных пакетов (не отброшенных);
- bdepartures_ — общее число байт, содержащееся в отправленных пакетах (не отброшенных);
- pdrops_ — общее число отброшенных пакетов;
- bdrops_ — общее число отброшенных байт;
- bytesInt_ — объект типа Integrator, который вычисляет интегральное значение очереди в байтах. Параметр sum_ содержит сумму (интеграл) размера очереди в байтах;

– `pktsInt_` — объект типа Integrator, который вычисляет интегральное значение очереди в пакетах. Параметр `sum_` содержит сумму (интеграл) размера очереди в пакетах.

Объекты - Агенты

– `$agent port` — возвращает порт транспортного уровня для агента;
– `$agent dst-addr` — возвращает адрес узла назначения, с которым соединён данный агент;
– `$agent dst-port` — возвращает порт узла назначения, с которым соединён данный агент;
– `$agent attach-source type` — устанавливает источник данных типа `type` (см. соответствующие методы агентов для информации о конфигурационных параметрах). Возвращает ссылку на объект источник;
– `$agent attach-traffic traffic-object` — прикрепляет объект `traffic-object` к агенту, который представляет собой генератор трафика `Traffic/Expoo`, `Traffic/Pareto` или `Traffic/Trace`.
`Traffic/Expoo` генерирует трафик, основанный на экспоненциальном On/Off распределении; `Traffic/Pareto` — на Парето On/Off распределении; `Traffic/Trace` — на основе trace файла.
– `$agent connect addr port` — соединяет агент `agent` с агентом, имеющим адрес `addr` и порт `port`. Это приводит к тому, что пакеты, отправляемые данным агентом, содержат информацию об адресе и порте, показывающую, что они должны быть направлены соответствующему агенту. Два данных агента должны быть совместимы (т.е. tcp-source/tcp-sink пара может соответствовать паре cbr/tcp-sink).

Конфигурационные параметры:
– `dst_` — адрес назначения, с которым соединён данный агент. Обычно состоит из 32 бит, из которых 24 бита определяют идентификатор узла назначения, а оставшиеся 8 бит — номер порта.

Null объекты

Null объекты — подкласс объектов агентов, являющихся приёмниками трафика. Они наследуют все функциональные особенности данных объектов. Null объекты не имеют никаких методов, конфигурационных параметров и переменных состояния.

LossMonitor объекты

LossMonitor объекты — подкласс объектов агентов, являющихся приёмниками трафика, которые также поддерживают сбор статистики о полученных данных, т.е. число полученных байт, число потерянных пакетов и т.д. Они наследуют все функциональные особенности объектов приёмников трафика:
– `$lossmonitor clear` — сбрасывает ожидаемый порядковый номер пакета (sequence number) в -1.

Параметры состояния:
– `nlost_` — число потерянных пакетов;
– `npkts_` — число полученных пакетов;
– `bytes_` — число полученных байт;
– `lastPktTime_` — время получения последнего пакета;
– `expected_` — ожидаемый порядковый номер (sequence number) следующего пакета.

TCP-объекты

Конфигурационные параметры:
- `window_` — верхняя граница заявленного окна TCP-соединения;
- `maxcwnd_` — верхняя граница окна перегрузки (переполнения) TCPсоединения (для отмены ограничения устанавливается в 0);
- `windowInit_` — начальное значение окна переполнения для медленного старта;
- `windowOption_` — алгоритм, использующийся для управления окном переполнения;
- `windowThresh_` — постоянная экспоненциально усредняющего (сглаживающего) фильтра, используемого для вычисления `awnd` (применяется для исследования различных алгоритмов увеличения окна);
- `overhead_` — случайно распределённая переменная, используемая для задержки каждого выходного пакета. Метод состоит в добавлении случайных задержек в источнике для устранения фазовых эффектов (применяется только в версии tcp Tahoe, в tcp Reno не используется);
- `ecn_` — при установке в true указывает, что в дополнение к отбрасыванию пакетов при переполнении используется механизм явного уведомления о перегрузке (*Explicit Congestion Notification, ECN*), что позволяет использовать быстрый повтор передачи (Fast Retransmit) после `quench()` в соответствии с битом ECN;
- `packetSize_` — размер всех пакетов источника;
- `tcpTick_` — таймер TCP, используемый для расчёта *времени двойного оборота пакета (Round-Trip Times, RTT)* — время движения пакета до узла назначения плюс время движения подтверждения. По умолчанию установлен в нестандартную величину 100 ms;
- `bugFix_` — указатель (true/false) запрета механизма быстрой повторной передачи при потере пакетов в одном окне данных;
- `maxburst_` — максимальное число пакетов, которое может посылать источник в ответ на одно полученное подтверждение (ACK) (при отсутствии ограничения устанавливается в 0);
- `slow_start_restart_` — признак использования (1/0) механизма медленного старта;

Определяемые величины:
- `MWS` (Maximum Window Size) — константа, определяющая максимальный размер окна (в пакетах) (по умолчанию `MWS=1024` пакетам).

Переменные состояния:
- `dupacks_` — число дублирующих подтверждений (ACK), полученных после прихода последнего недублирующего подтверждения;
- `seqno_` — наибольший номер последовательности сегмента данных (sequence number) источника TCP;
- `t_seqno_` — текущий номер последовательности сегмента пакета;
- `ack_` — наивысшее значение из полученных подтверждений;
- `cwnd_` — текущее значение окна перегрузки;
- `awnd_` — текущее значение окна переполнения при использовании усреднения;
- `ssthresh_` — текущее значение порога медленного старта;
- `rtt_` — оценка значения `round-trip time`;
- `srtt_` — оценка сглаженного значения `round-trip time`;
- `rttvar_` — оценка среднего отклонения значений `round-trip time`;
- `backoff_` — экспоненциальная постоянная задержки для параметра `round-trip time`.

Объекты TCPSINK

TCPSink-объекты представляют собой подкласс объектов агентов, являющихся приёмниками TCP пакетов. Симулятор использует только однонаправленные TCP-соединения, в которых TCP-источник посылает пакеты данных, а приёмник — подтверждения (ACK пакеты). TCPSink-объекты наследуют все функциональные особенности их родительских объектов. Для этих объектов не определено никаких методов и переменных.

Конфигурационные параметры:

- `packetSize_` — размер в байтах всех используемых пакетов подтверждений;
- `maxSackBlocks_` — максимальное число блоков данных, которое может быть подтверждено в опции SACK. Этот параметр используется только подклассом объектов TCPSink/Sack1. Эта величина не может быть увеличена для TCPSink-объекта после того как объект создан (после создания TCPSink-объекта величина может быть уменьшена, но не увеличена).

CONSTANT BIT-RATE объекты

CBR объекты предназначены для генерации пакетов данных с постоянной битовой скоростью:

- `$cbr start` — команда источнику начать генерацию;
- `$cbr stop` — остановка источника.

Конфигурационные параметры:

- `interval_` — задержка между генерацией пакетов;
- `packetSize_` — размер в байтах всех пакетов источника;
- `random_` — параметр определяет, присутствует ли случайный шум в процессе генерации пакетов. Если значение `random_` ноль, то время между генерацией пакетов определяется параметром `interval_`, в противном случае временной промежуток выбирается случайным образом из интервала `[0.5·interval_, 1.5·interval_]`.

Объекты типа SOURCE

Объекты Source генерируют данные для пересылки TCP.

Объекты `SOURCE/FTP`:

- `$ftp start` — команда источнику Source/FTP сгенерировать `maxpkts_` пакетов;
- `$ftp produce n` — команда источнику незамедлительно сгенерировать n пакетов;
- `$ftp stop` — команда прикреплённому TCP агенту прекратить пересылку данных;
- `$ftp attach agent` — прикрепляет Source/FTP объект к агенту `agent`;
- `$ftp producemore count` — команда источнику Source/FTP сгенерировать дополнительно `count` пакетов.

Конфигурационные параметры:

- `maxpkts` — максимальное число пакетов, генерируемых источником.

`TELNET SOURCE` объекты используются для генерации отдельных пакетов с заданными интервалами. Если `interval_` не ноль, то времена между генерацией пакетов выбираются из экспоненциального распределения со средним `interval_`. Если `interval_` имеет значение ноль, то времена между пакетами выбираются с использованием распределения tcplib telnet:

- `$telnet start` — запуск источника;
- `$telnet stop` — остановка источника;

– `$telnet attach agent` — прикрепление объекта Source/Telnet к агенту `agent`.
Конфигурационные параметры:
– `interval` — среднее время в секундах между пакетами, генерируемыми источником SOURCE/Telnet.

Объекты типа TRAFFIC

Объекты типа `Traffic` создают данные для пересылки по транспортному протоколу. Данные объекты могут быть прикреплены к агентам протокола UDP. Объекты типа `Traffic` создаются методами `Traffic/type`, где `type` `Expoo`, `Pareto` или `Trace`.

Объекты `Traffic/Expoo` генерируют On/Off трафик. В течение on-периодов пакеты генерируются с постоянной битовой скоростью. Во время off-периодов трафик не генерируется. Данные времена выбираются из экспоненциального распределения.

Конфигурационные параметры:
– `packet-size` — размер пакетов в байтах;
– `burst-time` — период генерации в секундах (on-период);
– `idle-time` — длительность off-периодов;
– `rate` — максимальная скорость в бит в секунду.

`TRAFFIC/PARETO` объекты аналогичны `Traffic/Expoo` объектам, за исключением того, что используется не экспоненциальное распределение Парето.

Конфигурационные параметры:
– `packet-size` — размер пакетов в байтах;
– `burst-time` — период генерации в секундах (on-период);
– `idle-time` — длительность off-периодов;
– `rate` — максимальная скорость в бит в секунду;
– `shape` — параметр формы Парето.

TRAFFIC/TRACE объекты

TRAFFIC/TRACE объекты используются для генерации трафика из trace файла:
– `$trace attach-tracefile tfile` — прикрепляет `Tracefile` объект `tfile` к `trace` объекту. `Tracefile` объект определяет trace-файл, из которого будут читаться данные трафика. К одному `Tracefile` объекту может быть прикреплено несколько `Traffic/Trace` объектов. Для каждого `Traffic/Trace` объекта выбирается случайное стартовое место в `Tracefile`.
Конфигурационные параметры для данного объекта не определены.

Tracefile объекты используются для определения trace файла, на основе которого будет генерироваться трафик
– `$tracefile` — создаёт объект `Tracefile`;
– `$tracefile filename trace-input` — устанавливает имя файла `filename`, из которого `trace` данные будут читаться в `trace-input`.

Методы TRACE и MONITORING

Объекты Trace используются для отслеживания действий в сети и записи этой информации в файл:
– `$ns create-trace type fileID node1 node2` — создаёт объект `Trace` типа `type` и прикрепляет к нему управляющую переменную `fileID` для мониторинга очереди между узлами `node1` и `node2`. Type может быть Enque, Deque или Drop. Enque отслеживает прибывающие в очередь пакеты. Deque — отправляемые пакеты, а Drop — отброшенные. FileID должна быть управляющей

файловой переменной, возвращаемой командой `Tcl open`. Соответствующий файл должен быть открыт для записи. Возвращает ссылку на trace объект.
- `$ns drop-trace node1 node2 trace` — удаляет trace объект, прикреплённый к соединению между узлами `node1` и `node2` с управляющей переменной `trace`.
- `$ns trace-queue node1 node2 fileID` — добавляет `Enque`, `Deque` и `Drop` мониторинг к соединению между узлами `node1` и `node2`.
- `$ns trace-all fileID` — добавляет `Enque`, `Deque` и `Drop Tracing` на все соединения топологии, созданные после вызова данной команды. Также добавляет отслеживание сетевой динамики. `FileID` должна быть управляющей файловой переменной, возвращаемой командой `Tcl open`. Соответствующий файл должен быть открыт для записи.
- `$ns monitor-queue node1 node2` — добавляет мониторинг длины очереди между узлами `node1` и `node2`. Возвращает объект типа `QueueMonitor`, который может быть использован для вычисления среднего размера очереди и т.д.
- `$ns flush-trace` — закрывает выходные каналы, прикреплённые ко всем trace объектам.
- `$link trace-dynamics ns fileID` — отслеживает динамику данного соединения и записывает данные в файл с управляющей переменной `fileID`; `ns` — переменная объекта типа `Simulator` или `MultiSim`, который был создан для обеспечения моделирования.

I.1.3. Файл трассировки

Трейс-файл представляет собой текст в формате ASCII, в котором зарегистрированы необходимые события моделирования (рис. I.1.1).

```
+ 0.02896 2 3 tcp 1040 ------- 2 4.0 3.1 2 3
- 0.02896 2 3 tcp 1040 ------- 2 4.0 3.1 2 3
+ 0.03 5 2 tcp 40 ------- 4 5.0 3.3 0 4
- 0.03 5 2 tcp 40 ------- 4 5.0 3.3 0 4
r 0.03096 2 3 tcp 1040 ------- 2 4.0 3.1 1 2
+ 0.03096 3 2 ack 40 ------- 2 3.1 4.0 1 5
- 0.03096 3 2 ack 40 ------- 2 3.1 4.0 1 5
```

Рис. I.1.1. Отрывок из trace-файла

Рассмотрим структуру трейс-файла (рис. I.1.2).

Событие	Время	От узла	К узлу	Тип пакета	Размер пакета	Флаги	Идент. потока	Адр. ист.	Адр. получ.	Поряд. номер	Идент. пакета

Рис. I.1.2. Структура trace-файла

- В поле «событие» (code) могут стоять символы:
  - `r`: receive — принятие пакета узлом;
  - `+`: enque — постановка в очередь;
  - `-`: deque — снятие с очереди;
  - `d`: drop — отбрасывание пакета из очереди;
  - `h`: hop — указывает на переход к следующему узлу.

- Поле «время» (time) показывает модельное время данного события в секундах.
- Поля: «от узла» (hsrc) и «к узлу» (hdst) показывают, в каком звене происходит данное событие.
- Поле «тип пакета» (packet type) указывает на то, к какому приложению или агенту относится данный пакет (tcp|telnet|cbr|ack и т.д.).
- Поле «размер пакета» (size packet) показывает размер пакета на сетевом уровне с учётом заголовка IP.
- Поле «флаги» (flags) содержит шесть флагов:
  - E — опытная индикация перегрузки;
  - N — индикация явного извещения о перегрузке с возможностью переноса;
  - C — эхо явного извещения о перегрузке;
  - A — сокращение окна перегрузки
  - P — приоритет (priority);
  - F — быстрый старт TCP (TCP Fast Start).
- Поле «идентификатор потока» (flowID) совпадает с полем fid (flow ID) в заголовке IPv6 (применяется для анализа сети, а также для использования разных цветов в визуализаторе nam)
- Поля «адрес источника» (src.sport) и «адрес получателя» (dst.sport) имеют формат `узел.порт` (например: 1.0).
- Поле «порядковый номер» (seq) показывает порядковый номер пакета на транспортном уровне.
- Поле «идентификатор пакета» (pktID) показывает уникальный идентификатор пакета.

I.1.4. NAM

В качестве средства анимации в NS-2 используется nam (Network Animator). Он графически воспроизводит имитационную модель (топологию сети, анимацию прохождения пакетов по сети, постановки их в очередь и т.д.) и наглядно показывает алгоритмы работы протоколов, дисциплин обслуживания очередей.

Запустить nam можно либо с помощью команды

```
nam <nam-file>
```

Здесь `<nam-file>` — имя трейс-файла nam, созданного ns2.

На рис. I.1.3 приведён интерфейс пользователя nam.

Интерфейс содержит зону анимации, несколько меню и кнопок.

В меню *Views* находятся следующие пункты:

- *New view* — создаёт новый вид той же анимации. Пользователь может увеличивать и прокручивать изображение в новом окне.
- *Show monitors* — показывает окно в нижней части экрана, где осуществляется мониторинг.
- *Show autolayout* — показывает окно в нижней части экрана, которое содержит окна для ввода данных и кнопки для настроек автоматического расположения.
- *Show annotation* – показывает пункт в нижней части экрана с примечаниями по мере увеличения модельного времени.

В меню *Analysis* находятся следующие пункты:

- *Active Sessions* — открывает окно со списком активных на данный момент сессий;
- *Legend ...* — открывает окно с описанием условных обозначений.

Меню *Help* содержит справочную информацию об аниматоре.

Под панелью меню находятся кнопки *перемотки назад*, *запуска анимации в обратную сторону*, *остановки анимации*, *запуска анимации*, *перемотки вперёд* и *выхода* из nam, *индикатор текущего времени анимации* и *движок изменения скорости*

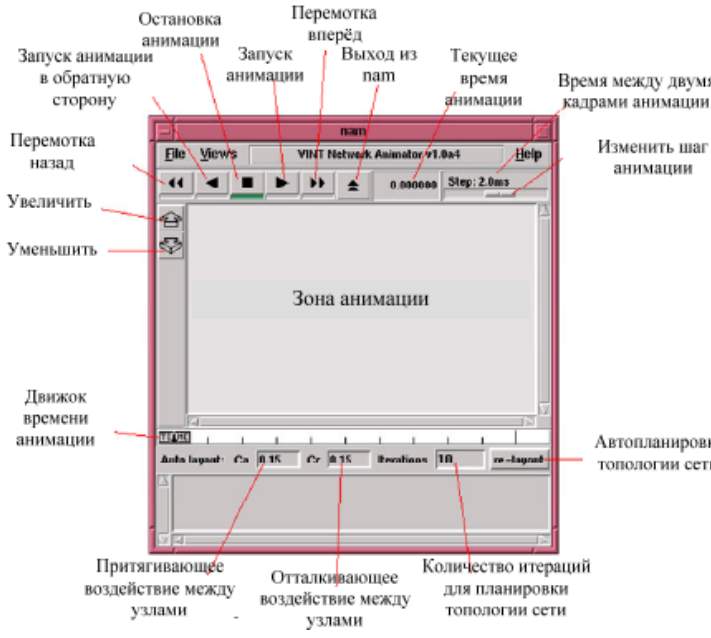

Рис. I.1.3. Интерфейс пользователя nam

анимации (текущая скорость изображена над ним). Также возможно увеличение и уменьшение изображения с помощью кнопок, расположенных в левой части экрана. Изменять текущее время анимации пользователь может при помощи *движка времени анимации*. Под движком времени анимации находится *панель автопланировки топологии сети* (изначально может отсутствовать).

Существует три параметра для настройки процесса автоматической планировки:

– *Ca* — константа притягивающего воздействия между узлами, которая контролирует силу сжатия между узлами в зоне анимации.
– *Cr* — константа отталкивающего воздействия между узлами, которая контролирует силу отталкивания между узлами в зоне анимации.
– *Количество итераций* — определяет, сколько раз запускать процедуру автопланировки.

Для маленьких топологий с десятками узлов использование исходных параметров (с 20–30 итерациями) достаточно для изображения приемлемого вида сети. Но для больших топологий необходимо изменение этих параметров внутри скрипта ns-2 предназначенными для этого командами.

I.1.5. Основы работы в Xgraph

Xgraph разработан для построения графиков в среде X-Windows. Xgraph способен читать данные из файлов или непосредственно со стандартного ввода. Он может отображать до 64 независимых наборов данных, используя различные цвета и/или

различные стили линий для каждого набора. При отображении графиков доступен вывод заголовков, меток осей, линий разметки или специальных отметок и условных обозначений.

Интерфейс, используемый для определения размера и расположения окна, зависит от используемого менеджера X-Windows. После того как окно было создано, все наборы данных отображаются графически с условными обозначениями в верхнем правом углу окна. Для увеличения части отображаемого графика нужно выделить её в окне xgraph, после чего она автоматически будет отображена в новом окне. Xgraph также имеет три кнопки управления в верхнем левом углу каждого окна: Close, Hardcopy и About.

Формат вызова Xgraph:

```
xgraph [ options ] [[-geometry |=]WxH+X+Y ] \
  [ -display host:display.screen ] [ file ... ]
```

Некоторые опции Xgraph:

- `-geometry WxH+X+Y или \=WxH+X+Y (Geometry)` — определение начального положения и размера окна xgraph;
- `-device [name]` — установка выходного устройства для xgraph (по умолчанию установлено 'X', другие доступные устройства — ps, hpgl, idraw и tgif);
- `-fitx` — масштабирование x-координат всех наборов данных к промежутку [0..1];
- `-fity` — масштабирование y-координат всех наборов данных к промежутку [0..1];
- `-scale [factor]` — выходной масштабный множитель для устройств postscript, hpgl и idraw. По умолчанию 1.0 и, например, при установке в 0.5 будет сгенерировано изображение с размером 50 % от исходного;
- `-fmtx [printf-format] -fmty [printf-format]` — устанавливает определенный формат отображения x или y осей;
- `-bb (BoundBox)` — отрисовка прямоугольников вокруг отображаемых данных. Полезно в случае использования меток вместо линий для отображения данных (см. опцию `-tk`);
- `-bd [color] (Border)` — определяет цвет границ для окна xgraph;
- `-bg [color] (Background)` — определяет цвет фона для окна xgraph;
- `-bw [size] (BorderSize)` — ширина границы окна xgraph;
- `-db (Debug)` — запуск xgraph в синхронном режиме и отображение значений всех величин, установленных по умолчанию;
- `-fg [color] (Foreground)` — установка цвета, которым отображаются все линии и текст в xgraph;
- `-gw (GridSize)` — ширина в пикселях линий разметки;
- `-gs (GridStyle)` — задание стиля отображаемых линий разметки;
- `-lf [fontname] (LabelFont)` – задание шрифта меток.
- `-lnx (LogX)` — отображение оси X в логарифмическом масштабе (разметка оси отображает степени десяти);
- `-lnx (LogY)` — отображение оси Y в логарифмическом масштабе (разметка оси отображает степени десяти);
- `-lw width (LineWidth)` — определяет ширину линий отображения данных в пикселях.
- `-lx [xl,xh] (XLowLimit, XHighLimit)` — опция ограничивает диапазон оси X определённым интервалом. Вместе с опцией `-ly` используется для масштабирования нужных участков больших графиков;
- `-ly [yl,yh] (YLowLimit, YHighLimit)` — опция ограничивает диапазон оси Y определённым интервалом.

– `-m (Markers)` — отметка каждой точки данных заданным маркером. В xgraph имеется восемь типов маркеров. Каждый тип имеет определенный вид линий и цвет.
– `-M (StyleMarkers)` — аналогично опции `-m`, но отдельный маркер присваивается каждому набору данных.
– `-nl (NoLines)` — опция отключает отображение линий. Вместе с опциями -m, -M, -p, или -P используется для отображения точечных графиков;
– `-ng (NoLegend)` — отключение отображения условных обозначений;
– `-p (PixelMarkers)` — маркирование каждой точки данных отдельным небольшим маркером (пиксельного размера). Обычно используется вместе с опцией `-nl` для построения точечных графиков;
– `-P (LargePixels)` — аналогично `-p`, но используются крупные маркеры;
– `-t [string] (TitleText)` — определение заголовка графика (строка-заголовок будет отображена в центре сверху графика);
– `-tk (Ticks)` — отображение данных с помощью маркеров, а не линий;
– `-tkax (Tick Axis)` — отображает оси при использовании маркеров;
– `-x [unitname] (XUnitText)` – определяет подпись к оси X;
– `-y [unitname] (YUnitText)` — определяет подпись к оси Y;
– `-zg [color] (ZeroColor)` — определяет цвет, используемый для отображения нулевой осевой отметки;
– `-zw [width] (ZeroWidth)` — ширина нулевой осевой отметки в пикселях.

I.1.6. Основы работы в Gnuplot

Gnuplot — программа для построения графиков функций и визуализации различных данных.

Работа в Gnuplot возможна в двух режимах:
– пакетном — готовится специальный файл, содержащий последовательность команд;
– интерактивном — обращение к программе осуществляется через командную строку в режиме реального времени.

Базовые команды Gnuplot:
– `help` — вывести справку;
– `load <имя файла>` — загрузить командный файл.

Терминалом в gnuplot является то устройство (или файл), в которое будет осуществляться вывод полученного результата. Таковым может быть монитор, принтер или же файл с расширением png, jpg, eps и др., а также latex-файл. Тип терминала задаётся командой:

```
set terminal <тип терминала>
```

Здесь `<тип терминала>` может принимать следующие значения:
– `windows` — вывод данных на дисплей в ОС Windows;
– `X11` — вывод данных на дисплей в ОС Linux;
– `png` — вывод данных в файл формата png (растровый формат);
– `jpeg` — вывод данных в файл формата jpeg (растровый формат);
– `postscript eps` — вывод данных в файл формата eps (векторный формат);
– `latex` — вывод данных в файл формата LaTex.

Пример вывода в файл:

```
#устанавливаем тип терминала
set terminal postscript eps

#устанавливаем имя выходного файла
set output "plot1.eps"
```

Для построения графика функции на плоскости используется команда `plot` (знак «\» обозначает переход на другую строку):

```
plot [<изменение аргумента>] [<изменение функции>] \
     <функция> <доп. параметры>
```

Например, график синусоиды при изменении x от -2π до 2π:

```
plot [-2*pi:2*pi] sin(x)
```

Область изменения значений аргумента/функции использует команды:

```
set xrange [<нач. значение>:<конечн. значение>]
set yrange [<нач. значение>:<конечн. значение>]
```

При выводе Gnuplot позволяет устанавливать различные визуальные параметры для графика. Для этого в команде `plot` после объявления функции следует ввести:

```
with <стиль графика> \
linetype <тип, целое число (комбинация стиль+цвет)>
```

Некоторые простые стили:
- `lines` (по умолч.) — линии;
- `points` — точки;
- `lines and points` — линии с точками;
- `dots` — очень маленькие точки;
- `impulses` — дискретные прямые;
- `steps` — ломаная под прямым углом линия.

Для изменения цвета и толщины линии графика в команде `plot` нужно указать:

```
linestyle <цвет, целое число> \
linewidth <толщина линии, pt>
```

Приведём пример построения графика функции $\sin(x)$:

```
#!/usr/bin/gnuplot -persist

# вывод в eps-файл
set terminal postscript eps enhanced color

# назначаем выходной файл
set output "plot1.eps"

# устанавливаем кодировку для кириллицы
set encoding utf8

# оси OX и OY, шрифт
set ylabel "ось y" font "Helvetica,18"
set xlabel "ось x" font "Helvetica,18"

# отступы
set bmargin 4  # отступ снизу
set lmargin 10 # отступ слева
set rmargin 10 # отступ справа
set tmargin 4  # отступ сверху

# метки по осям Ox,Oy
set xtics ("10"0,"20"1,"30"2)
set ytics ("-30"-2,"-20"-1,"0"0,"20"1,"30"2)
```

```
# изменение по Ox, Oy
set xrange [-2*pi:2*pi]
set yrange [-2:2]

# построение, цвет, толщина
plot sin(x) with lines lt 3 lw 2
```

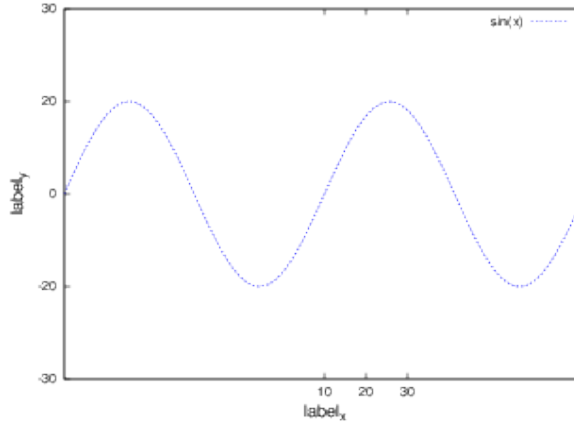

Рис. I.1.4. График функции $y = \sin(x)$

Для построения поверхностей в Gnuplot применяется команда `splot`. Обращение к ней происходит аналогично команде `plot`, за исключением некоторых особенностей.

Формировать данные для построения графиков можно не только задавая промежутки изменения переменных, но и используя заранее подготовленный файл с данными.

I.1.7. AWK

AWK — утилита, предназначенная для простых (механических и вычислительных) манипуляций над данными:

- использует метод поиска по шаблону (pattern matching);
- оперирует двумя видами входных данных: файлом данных и командным;
- файл данных — упорядоченные данные, состоящие из строк, которые в свою очередь состоят из групп знаков (слов), разделённых пробелами;
- командный файл — инструкции (команды) поиска по шаблону;
- AWK — интерпретатор, исполняющий действия, записанные в командном файле, над файлом данных;
- все команды одновременно могут использовать все переменные программы AWK и только одну строку из файла данных — она автоматически загружается в специальные переменные;

– переменная `$0` содержит всю строку, `$1` — первое слово в строке, `$2` — второе слово и т.д. (максимальное количество — 100 слов).

Пример кода AWK, вычисляющий среднее значение чисел, записанных в четвёртой колонке:

```
BEGIN {FS =" "}{nl++} {s=s+$4} END {print "average:" s/nl}
```

Список литературы

1. *Заборовский В. С.* Моделирование и анализ сетей связи с коммутацией пакетов. Network Simulator (Сетевой симулятор ns2). — СПб : Изд-во СПбГТУ, 2001. — 108 с.
2. Exercises on ”ns-2” / C. Barakat. — Заявл. 2003.
3. *Галкин А. М.*, *Кучерявый Е. А.*, *Молчанов Д. А.* Пакет моделирования NS-2: учеб. пособие. — СПб : СПбГУТ, 2007.
4. *Заборовский В. С.*, *Мулюха В. А.*, *Подгурский Ю. Е.* Моделирование и анализ компьютерных сетей: телематический подход. — СПб : Изд-во СПбГПУ, 2010. — 93 с.