Федеральное агентство по образованию
Государственное образовательное учреждение
высшего профессионального образования
Владимирский государственный университет
Кафедра физики и прикладной математики

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам
по курсу «Компьютерное моделирование»

Составитель
П.Ю. ШАМИН

Владимир 2010

УДК 00494
ББК 32.97
М54

Рецензент
Доктор технических наук, профессор,
заведующий кафедрой информатики и защиты информации
Владимирского государственного университета
М.Ю. Монахов

Печатается по решению редакционного совета
Владимирского государственного университета

Методические указания к лабораторным работам по курсу
М54 «Компьютерное моделирование» / Владим. гос. ун-т ; сост.
П. Ю. Шамин. – Владимир : Изд-во Владим. гос. ун-та, 2010. – 45 с.

Методические указания ориентированы на выполнение лабораторных работ с использованием параллельного сетевого симулятора (ПСС). Описывается назначение, характеристики и правила применения ПСС. Приводятся примеры использования ПСС, а также даются задания к лабораторным работам с его применением.

Предназначены для студентов специальности 010503 «Математическое обеспечение и администрирование информационных систем» для использования в рамках курсов «Архитектура вычислительных систем и компьютерных сетей» или «Компьютерное моделирование» дневного обучения 4-го курса, также могут быть полезны студентам других информационных специальностей.

Табл. 2. Ил. 2. Библиогр.: 3 назв.

УДК 00494
ББК 32.97

Введение

Настоящие методические указания описывают назначение, характеристики и правила применения параллельного сетевого симулятора (ПСС). Данный программный комплекс позволяет производить моделирование сетей различных конфигураций, в том числе гетерогенных и с переменной топологией. Приведены возможные варианты лабораторных работ с использованием ПСС (Прил. 1).

Моделирование выполняется с использованием технологий параллельных вычислений на SMP-системе или суперкомпьютере кластерной архитектуры. В случае необходимости моделирование с использованием ПСС может производиться на обыкновенном персональном компьютере.

Система ПСС позволяет генерировать топологию сети (в том числе большой размерности – сотни тысяч узлов), динамически её переопределять, имитировать движение узлов, а также их включение и выключение, имитировать работу различных сетевых протоколов.

С помощью данного программного обеспечения можно проводить ознакомление с устройством и принципами функционирования сетей различных типов, функционированием алгоритмов маршрутизации и т. д. Кроме того, программа может быть полезна для изучения собственно технологий компьютерного моделирования.

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПСС

1.1. Назначение параллельного сетевого симулятора

Основным назначением параллельного сетевого симулятора ПСС является моделирование работы сетей с переменной топологией с целью проверки корректности функционирования управляющего программного обеспечения узлов, сетевых протоколов, алгоритмов маршрутизации в тех или иных условиях. Однако, при соответствующем подборе модулей расширения, возможно моделирование с помощью ПСС и других сетевых структур практически любой природы.

1.2. Условия применения ПСС

1.2.1. Требования к программному обеспечению

Для корректного функционирования ПСС на каждом узле кластерной вычислительной системы, используемой для запуска ПСС, должно быть установлено следующее программное обеспечение:

– операционная система семейства Linux с ядром версии 2.6.x или новее;

– библиотека поддержки параллельного программирования в стандарте MPI (например, mvapich);

– комплект драйверов, необходимых для поддержки функционирования сетевого интерфейса (Ethernet, Infiniband и т. п.), обеспечивающего взаимодействие между узлами вычислительной системы с распределённой памятью;

Для разработки модулей расширения ПСС на компьютере разработчика в дополнение к вышеперечисленному программному обеспечению должен быть установлен компилятор GNU C++ (GCC) версии 4.3.x или новее.

1.2.2. Требования к оборудованию

Для корректного и эффективного функционирования ПСС требует наличия вычислительной системы со следующими характеристиками:

– один или более вычислительных узлов с процессорами той архитектуры, для которой была осуществлена сборка ПСС и модулей расширения, средствами компилятора языка C++;

– сетевой интерфейс[1] с пропускной способностью, достаточной для обеспечения эффективной передачи данных между узлами вычислительной системы в ходе симуляции средствами MPI. Для данной версии ПСС достаточно способности сети Ethernet с пропускной способностью 10 Мбит/с;

– оперативная память, достаточная для размещения данных об узлах симулируемой сети и текущих служебных данных подключаемых модулей, а также служебных буферов ядра ПСС. В общем случае объём оперативной памяти, необходимый для выполнения моделирования увеличивается с ростом числа узлов моделируемой сети и сложности моделируемых сетевых протоколов;

– устройство долговременного хранения данных, доступное с узла вычислительной системы, на котором функционирует управляющий экземпляр ПСС. Оно необходимо для сохранения файла журнала текущего сеанса моделирования и должно иметь свободный объём, достаточный для сохранения данного файла (точная величина зависит от эксперимента).

1.2.3. Прочие требования и ограничения

Для корректного функционирования журнала сеанса моделирования ПСС пользователь, от имени которого выполняется запуск ПСС, должен иметь право на запись в файл журнала сеанса моделирования ПСС.

Возможность дублирования вывода результатов моделирования и служебных сообщений ПСС на экран, включаемая установкой опции log_to_console в конфигурационном файле simultor.conf в отличное от нуля значение, требует наличия связи потока стандартного вывода процесса ПСС с консольным окном, в которое будет производиться вывод сообщений.

В случае аварийного завершения ПСС, например принудительным завершением процесса ПСС административными средствами операционной системы, корректное сохранение результатов моделирования в файл журнала сеанса моделирования ПСС не гарантируется.

---

[1] Сетевой интерфейс не требуется, если запуск ПСС выполняется на одном узле вычислительной системы в многопоточном режиме с целью задействовать возможности многоядерного процессора.

2. ХАРАКТЕРИСТИКА ПРОГРАММЫ

Для обеспечения максимальной универсальности ПСС, в основу его архитектуры положен принцип модульности. В соответствии с этим принципом ПСС и состоит из ядра и модулей расширения, выбор которых позволяет конструировать нужную модель и измерять на ней нужные параметры. При этом функционал ПСС разделён между ядром и модулями расширения.

В число основных функций ядра ПСС входят следующие:

– загрузка, конфигурация, вызов и выгрузка модулей расширения;
– хранение модели топологии сети;
– отсчёт модельного времени;
– транспортировка виртуального трафика;
– сбор и сохранение информации о работе сети, накопленной модулями расширения;
– отслеживание условия завершения моделирования и корректное завершение моделирования.

Функциональные возможности модулей расширения определяются разработчиками модулей расширения и в общем случае могут состоять в следующем:

– генерация топологии виртуальной сети;
– модификация топологии виртуальной сети;
– генерация виртуального трафика;
– фильтрация виртуального трафика;
– определение путей следования виртуального трафика;
– учёт виртуального трафика;
– определение момента завершения моделирования;
– прочие функции.

ПСС может применяться для моделирования сетей различной размерности и на различном уровне детализации. Однако, моделирование сетей большой размерности (миллионы узлов) может потребовать больших объёмов оперативной памяти (несколько гигабайт) ввиду того, что текущей версией ПСС не поддерживается посегментное моделирование топологии сети на различных узлах кластерных вычислительных систем. Данная функция будет реализована в последующих версиях ПСС.

3. ОБРАЩЕНИЕ К ПРОГРАММЕ

Запуск ПСС штатным образом выполняется с использованием средств управления заданиями из состава программного обеспечения параллельной вычислительной системы, на которой он запускается. В качестве исполняемого файла при этом указывается командный файл операционной системы, содержащий вызов утилиты mpirun из состава MPI-библиотеки, установленной на вычислительной системе. В параметрах mpirun в качестве исполняемого файла указывается исполняемый файл ПСС – Sim.

Передача входных данных (в том числе параметров моделирования) производится с использованием файлов конфигурации. Подробное описание структуры входных данных приводится ниже в разделе «Входные и выходные данные».

4. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

4.1. Входные данные

4.1.1. Способ передачи входных данных

Передача входных данных ядру ПСС и модулям расширения ПСС осуществляется с использованием текстовых конфигурации. Так как эти файлы текстовые, то для их редактирования может быть использован любой текстовый редактор. Ядро ПСС использует два файла конфигурации – simulator.conf и modules.conf. Для нормальной работы ПСС достаточно наличия файлов конфигурации ядра ПСС на узле вычислительной системы, на котором запускается управляющий процесс ПСС.

4.1.2. Конфигурационный файл simulator.conf

Конфигурационный файл simulator.conf предназначен для хранения различных настроек ядра ПСС. Чтение настроек ядра ПСС из файла simulator.conf производится при запуске процесса моделирования. Файл simulator.conf является текстовым файлом и состоит из строк вида:

<имя_параметра> <значение параметра>

Поле <имя_параметра> должно содержать имя одного из параметров конфигурации, поддерживаемых данной версией ядра ПСС. Если какие-то параметры ядра ПСС не заданы в конфигурационном файле simulator.conf, то используются их значения по умолчанию. Перечень основных параметров ядра ПСС, которые могут задаваться в файле simulator.conf и их значения по умолчанию приведены ниже в табл. 1.

Табл. 1. Параметры, задаваемые в simulator.conf

Наименование параметра	Значение по умолчанию	Назначение параметра	Примечание
sim_net_peers	5000	Число узлов моделируемой сети	
sim_max_simulation_steps	0	Предельное число шагов моделирования	Если 0 – не лимитировано
log_to_file	1	Вывод результатов моделирования в файл	Если отлично от нуля – вывод производится
log_to_console	1	Вывод результатов моделирования в консоль	Если отлично от нуля – вывод производится
log_file_name	sim.log	Имя файла результатов моделирования	Если имя файла без пути или путь относительный – при определении полного пути используется рабочий каталог ПСС
peer_packet_buf_size	50	Размер буферов для входящих и исходящих пакетов, имеющихся на симулируемых узлах	Пакеты, не поместившиеся в буфер, отбрасываются. В файл журнала сеанса моделирования выдаётся предупреждающее сообщение
peer_stat_buf_size	256	Размер буфера статистики, имеющегося на симулируемых узлах	Данные, не поместившиеся в буфер, отбрасываются. В файл журнала сеанса моделирования выдаётся предупреждающее сообщение
peer_max_neighbors	50	Максимальное количество смежных узлов для симулируемого узла	Связи, не поместившиеся в буфер, не добавляются. В файл журнала сеанса моделирования выдаётся предупреждающее сообщение
sim_allow_teleportation	0	Разрешение отправки пакета несмежному узлу	Разрешение отправки пакетов несмежным узлам путём установки данного параметра в отличное от нуля значение несколько повышает быстродействие за счёт отключения контроля отправки пакетов ядром ПСС

В текущей версии ПСС параметры, заданные в simulator.conf, но отсутствующие в данной таблице, игнорируются.

4.1.3. Конфигурационный файл modules.conf

Конфигурационный файл modules.conf предназначен для указания перечня подключаемых модулей, используемых при моделировании и определения порядка их загрузки и вызова. Чтение этих данных из файла simulator.conf производится при запуске процесса моделирования. Файл modules.conf является текстовым файлом и состоит из строк вида:

<имя_модуля>

Поле <имя_модуля> должно содержать имя динамической библиотеки модуля расширения ПСС (без стандартного префикса «lib» и окончания «.so»). Например, для модуля расширения Module1, размещённого в динамической библиотеке libModule1.so следует указать «Module1». Этот модуль должен либо присутствовать в текущем каталоге, которым в данном случае является рабочий каталог ПСС, либо находиться в одном из каталогов, перечисленных в переменной среды LD_libery_Path.

Порядок загрузки, инициализации, вызова и финализации модулей расширения ядром ПСС соответствует порядку их указания в файле modules.conf. В случае невозможности загрузки модуля расширения, указанного в файле modules.conf, ядро ПСС преждевременно завершает свою работу с выдачей соответствующего сообщения в файл журнала сеанса моделирования ПСС.

4.1.4. *Конфигурационные файлы модулей расширения*

Параметры конфигурации модулей расширения ПСС задаются с помощью конфигурационных файлов модулей расширения ПСС. Структура и способ использования содержимого конфигурационных файлов модулей расширения ПСС определяются разработчиками модулей расширения ПСС.

Файл модуля расширения должен иметь имя вида <имя_модуля_расширения>.conf. В этом случае ядро ПСС обеспечивает передачу содержимого файла конфигурации модуля расширения ПСС этому модулю при его инициализации (подробнее см. раздел 9 «Интерфейс модуля расширения ПСС»).

В текущей версии модуль расширения должен уметь самостоятельно интерпретировать структуру своего модуля расширения. В последующих

версиях ПСС будет реализован централизованный механизм получения модулями расширения параметров конфигурации из своих файлов конфигурации, вследствие чего данное ограничение будет преодолено.

4.2. Выходные данные

Выходными данными ПСС являются служебные сообщения ядра ПСС, например о возникающих в ходе моделирования ошибках, данные о ходе моделирования, собранные модулями расширения ПСС. Сообщения об ошибках представляют собой текстовые сообщения, содержащие информацию о произошедшем сбое и, возможно, о путях решения проблемы. Состав выходных данных, формируемых модулями расширения, определяется разработчиками модулей расширения.

Выходные данные ПСС, сохраняются в файле журнала сеанса моделирования ПСС. По умолчанию файлом журнала сеанса моделирования ПСС является файл sim.log, однако можно задать любой другой с помощью изменения параметра log_file_name в конфигурационном файле simulator.conf. Выходные данные модулей расширения также сохраняются в файле журнала сеанса моделирования ПСС. Так как файл журнала сеанса моделирования ПСС является текстовым файлом, то для его просмотра можно использовать любой текстовый редактор. Исключением является случай, когда разработчик модуля расширения ПСС решил использовать в выходных данных своего модуля бинарное представление данных. В этом случае часть данных в файле журнала окажется бинарными, что потребует использования для его просмотра специализированного программного обеспечения.

5. СООБЩЕНИЯ

Ядро ПСС в ходе моделирования может формировать различные сообщения отладочного характера, в том числе и являющиеся сообщениями об ошибках. Эти сообщения выводятся в файл журнала сеанса моделирования ПСС и, при соответствующей настройке, в консольное окно.

Сообщения ядра ПСС об ошибках могут быть идентифицированы по префиксу «Fatal error» (для фатальной ошибки) или «Non-fatal error» (для нефатальной ошибки) сообщения. После префикса в сообщении указывается информация о возникшей проблеме. Как правило, возникновение фатальных ошибок при работе ядра ПСС приводит к немедлен-

ному завершению процесса моделирования, поэтому при возникновении такой ситуации следует найти и устранить причину, её вызвавшую.

Следует иметь в виду, что сообщения с префиксом «Non-fatal error» содержат информацию об ошибке в ходе моделирования, влияющей на его результат, но не опасной для функционирования ядра ПСС и не сопровождаются немедленным завершением процесса моделирования.

6. АРХИТЕКТУРА ПСС

Основной особенностью ПСС, являющейся залогом его высокой универсальности, является обеспечение расширяемости симулятора за счёт унифицированной архитектуры модулей расширения. Это в частности позволяет минимизировать время, необходимое на разработку и реализацию на основе данного симулятора специализированных систем симуляции для проведения различных экспериментов. В частности, добавление и изменение состава используемых модулей расширения не потребует перекомпиляции ядра симулятора.

В архитектуре ПСС собственно симулятор представляет собой базовый элемент, обеспечивающий сервисные функции (транспортировку пакетов между узлами, сохранение статистики), а вся остальная функциональность реализована с помощью подключаемых модулей (модулей расширения), имеющих унифицированный интерфейс. Графическая иллюстрация данной концепции приведена на рис. 1.

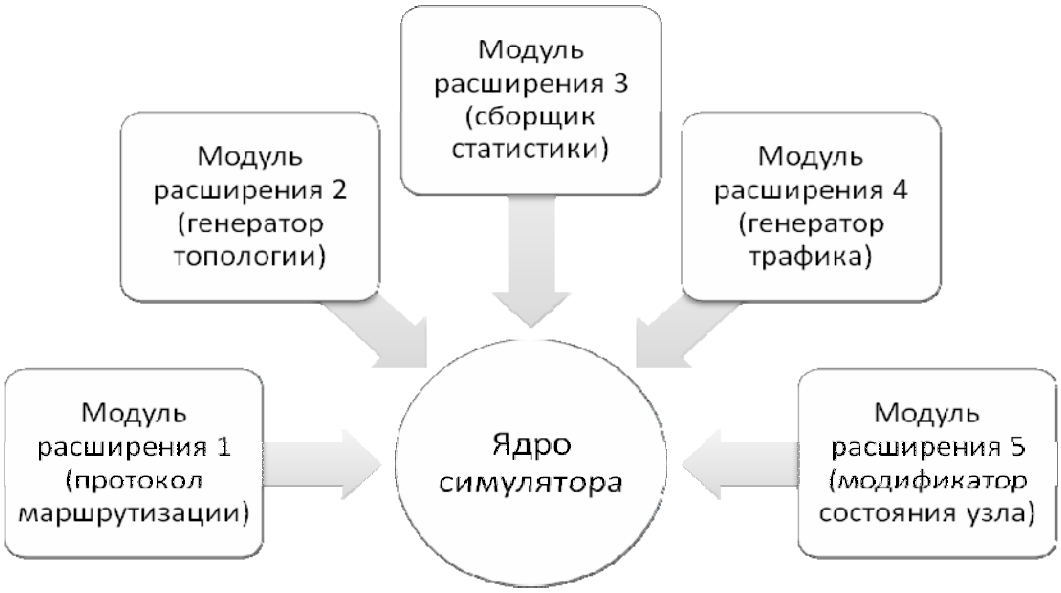


Рис. 1. Модульная архитектура ПСС

При этом возможно подключение к ПСС произвольного количества модулей расширения различных типов. Все они получают управление по очереди и им передаются данные для обработки.

Можно выделить следующие типы модулей расширения:

– протокол маршрутизации;
– генератор топологии;
– генератор трафика;
– модификатор (генератор) состояния узла;
– модификатор (генератор) состояния симулятора;
– сборщик статистики работы узла;
– прочие модули расширения.

При необходимости один модуль расширения может выполнять функции модулей нескольких типов, в этом случае получается комбинированный модуль расширения. Типовой набор модулей расширения представлен в Прил. 2.

7. АЛГОРИТМ ФУНКЦИОНИРОВАНИЯ ПСС

Архитектура ПСС рассчитана на работу с дискретным временем, поэтому работа ПСС происходит циклически. Процесс моделирования завершается по некоторому условию. В качестве такого условия может выступать:

– выполнение заданного количества шагов симуляции;
– отсутствие пересылок пакетов на последнем шаге симуляции (или за заданное количество последних шагов);
– накопление необходимого объёма статистики;
– наступление иного события (например, достижение заданным пакетом заданного узла).

С момента запуска процесса моделирования и до выполнения условия завершения моделирования ядро ПСС циклически повторяет для каждого узла симулируемой сети[2] следующую последовательность действий:

---

[2] Здесь и далее термином «узел симулируемой сети» или просто «симулируемый узел» обозначен некий виртуальный объект, являющийся программной моделью узла сети передачи данных, симуляция которой выполняется. В противоположность ему, терминами «узел кластера» или «вычислительный узел» обозначен вполне реальный объект – процессор, ядро процессора или целый компьютер, являющийся частью вычислительной системы, на которой производится симуляция.

1. Последовательный вызов модулей расширения.
2. Транспортировка пакетов данных, ожидающих отправки.
3. Транспортировка статистики.

Рассмотрим подробнее каждую из этих задач.

Последовательный вызов модулей расширения производится каждым вычислительным узлом для каждого симулируемого им узла. Вызов производится вызовом метода DoWork()[3] модуля расширения с передачей ему структуры данных соответствующего симулируемого узла.

Транспортировка пакетов данных, ожидающих отправки, производится каждым вычислительным узлом для каждого узла симулируемой сети в следующем порядке:

1. Очистка буферов входящих пакетов в структуре данных каждого симулируемого узла.

2. Перенос пакетов данных из буфера исходящих пакетов каждого симулируемого узла в буфер входящих пакетов соответствующего симулируемого узла. При этом если симулируемый узел обсчитывается другим вычислительным узлом[4], перенос происходит в два этапа: отправка одним вычислительным узлом управляющего сообщения другому (какому симулируемому узлу и что передать) и собственно помещение пакета данных вычислительным узлом, получившим управляющее сообщение в буфер входящих пакетов соответствующего симулируемого узла. Данный механизм требует установления тем или иным образом соответствия между номером узла вычислительной сети и набором идентификаторов симулируемых узлов, обсчитываемых данным вычислительным узлом, что также осуществляется ядром симулятора.

Транспортировка статистики представляет собой отправку каждым вычислительным узлом в конце каждого шага симуляции содержимого своего буфера статистики для каждого симулируемого им узла выделенному вычислительному узлу с целью сохранения. Затем вычислительный узел производит очистку буфера статистики для каждого симулируемого им узла.

Перед запуском основного цикла работы ПСС, описанного выше, ядро ПСС выполняет загрузки и инициализацию модулей расширения.

---

[3] См. раздел 9 «Интерфейс модуля расширения».

[4] В текущей версии ПСС не реализовано.

Инициализация модулей расширения производится последовательным вызовом метода Init()[8] каждого модуля расширения с передачей ему содержимого файла конфигурации модуля расширения.

После завершения основного цикла работы ПСС ядро ПСС выполняет финализацию работы модулей расширения последовательным вызовом метода Shutdown()[5] для каждого модуля расширения, при этом производится сборка финальной статистики работы модуля расширения.

Блок-схема алгоритма работы ПСС приведена в прил. 3 к данным методическим указаниям.

8. СТРУКТУРА ДАННЫХ СИМУЛИРУЕМОГО УЗЛА

Для каждого узла симулируемой сети симулятор хранит структуру данных, характеризующую его состояние на текущий момент. Фактически эта структура данных представляет собой программную модель узла симулируемой сети. Её состав приведён на рис. 2.

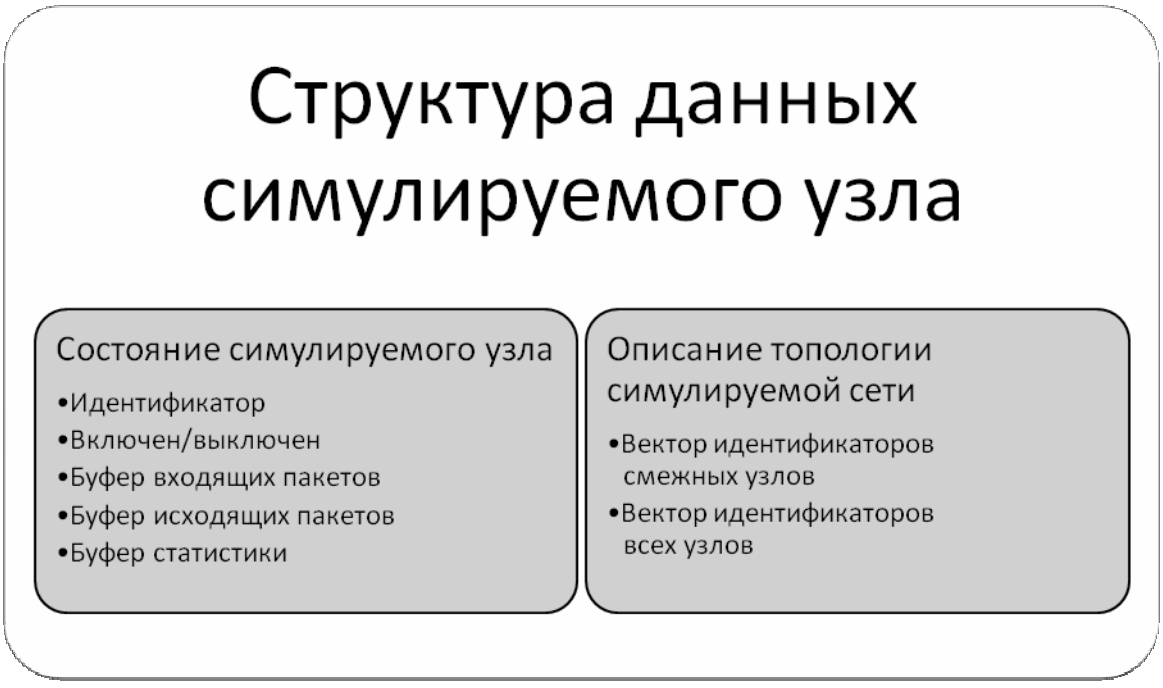


Рис. 2. Структура данных симулируемого узла

[5] См. раздел 9 «Интерфейс модуля расширения ППС».

9. ИНТЕРФЕЙС МОДУЛЯ РАСШИРЕНИЯ ПСС

Как указывалось выше, ПСС реализован по модульному принципу, причём ядро ПСС выполняет лишь некоторые базовые функции, поэтому для проведения с его помощью каких-либо экспериментов, требуется разработка соответствующих модулей расширения, реализующих требуемую логику модели.

Модули расширения ПСС представляют собой динамически загружаемые библиотеки с унифицированным интерфейсом, поэтому для создания собственных модулей расширения программисту не требуется исходный код ядра ПСС и каждый модуль расширения может разрабатываться и тестироваться независимо от других. Более того, при наличии всех необходимых модулей расширения, проведение совершенно другого эксперимента требует всего лишь редактирования конфигурационных файлов ядра ПСС (в первую очередь modules.conf) и, возможно, файлов конфигурации модулей расширения.

Интерфейс модуля расширения чрезвычайно прост и состоит всего из нескольких методов, которые могут быть вызваны ядром ПСС при использовании данного модуля расширения. Рассмотрим данные методы в контексте жизненного цикла модуля расширения.

Работа ПСС начинается с загрузки динамических библиотек модулей расширения. Необходимые библиотеки перечислены в конфигурационном файле modules.conf, доступном управляющему экземпляру ПСС. Файл modules.conf представляет собой текстовый файл, в котором перечислены все используемые в данном эксперименте модули расширения в том порядке, в котором предполагается их использование. Перед запуском процесса моделирования управляющий экземпляр ПСС рассылает остальным экземплярам перечень модулей подлежащих загрузке и содержимое их файлов конфигурации.

Затем производится загрузка динамических библиотек модулей расширения в память и их инициализация. Чтобы выполнить инициализацию модуля расширения ядро ПСС на каждом вычислительном узле находит и вызывает метод Init() каждого модуля расширения. Прототип этого метода выглядит следующим образом:

```
IExtensionModule * Init(char * configFile);
```

При вызове метода Init() модуль расширения производит специфическую для него инициализацию и возвращает указатель на интерфейс IExtensionModule[6], через который производится дальнейшая работа ядра симулятора с данным модулем расширения.

Параметр configFile содержит текст конфигурационного файла данного модуля расширения. Этой файл конфигурации должен иметь имя <имя_модуля_расширения>.conf. При отсутствии соответствующего файла конфигурации модулю передаётся NULL.

Интерфейс IExtensionModule содержит два метода, первый из которых предназначен для выполнения собственно основной работы модуля расширения. Этот метод называется DoWork(), вызывается ядром ПСС один раз на каждом шаге симуляции и имеет следующий прототип:

```
void DoWork(PeerState * peerState, SimState * simState);
```

Здесь PeerState[7] – класс, содержащий состояние текущего симулируемого узла (подробнее см. в разделе «Структура данных симулируемого узла»), а SimState[7] – класс, содержащий общие данные о состоянии симулятора в целом (например, номер цикла симуляции – аналог текущего времени). Передача значений по указателю обеспечивает отсутствие необходимости пересылки больших буферов данных.

Второй метод, поддерживаемый интерфейсом IExtensionModule, используется при завершении процесса моделирования. Это метод Shutdown(). Вызов этого метода выполняется ядром ПСС однократно для каждого модуля расширения при завершении процесса симуляции. При его вызове модуль выполняет некие специфические для него действия (например, очистку памяти) и возвращает (при необходимости) буфер с финальной статистикой своей работы. Метод имеет следующий прототип:

```
void Shutdown(int * bufSize, char ** buf);
```

Если модуль расширения не возвращает статистики, то bufSize и buf должны быть NULL. Если же статистика возвращается, то bufSize

---

[6] Класс, реализующий логику работы модуля расширения, должен быть унаследован от данного интерфейса.

[7] Соответствующие типы данных и интерфейсы описаны в файле SimDataTypes.h, который распространяется вместе с ПСС.

должен указывать её объём. В этом случае ядро ПСС пересылает статистику узлу сборщику статистики на управляющем экземпляре ПСС.

Рассмотрим подробнее изменения, производимые ядром ПСС в структуре PeerState или SimState при вызове метода DoWork() в зависимости от функционального назначения модуля расширения (табл. 2).

Табл. 2. Типы модулей расширения

Тип модуля расширения	Данные, подлежащие модификации
Протокол маршрутизации	Буфер исходящих пакетов структуры PeerState
Генератор топологии	Вектор идентификаторов смежных узлов структуры PeerState
Генератор трафика	Буфер исходящих пакетов структуры PeerState
Модификатор (генератор) состояния узла	Флаг «включен/выключен», другие флаги состояния узла структуры PeerState
Сборщик статистики работы узла	Буфер статистики структуры PeerState
Фильтр пакетов	Буфер входящих пактов структуры PeerState
Модификатор (генератор) состояния ПСС	Флаги состояния симулятора структуры SimState, например, флаг окончания симуляции
Комбинированный модуль	Одна или более частей структуры PeerState и/или SimState

10. ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ О ВНУТРЕННЕЙ АРХИТЕКТУРЕ И ФУНКЦИОНИРОВАНИИ ПСС

10.1. Библиотека поддержки модулей расширения

Для обеспечения поддержки в ПСС модулей расширения была создана специальная статическая библиотека SimDataTypes (файл libSimDataTypes.a). Эта статическая библиотека компонуется с каждой динамической библиотекой модулей расширения ПСС и обеспечивает их необходимыми средствами интеграции с ядром ПСС.

В библиотеку SimDataTypes вынесен только необходимый код, поэтому она имеет малый размер, и наличие её в каждом модуле расширения не приводит к значительному перерасходу оперативной памяти. Библиотека SimDataTypes имеет размер менее 8 Кб.

Исходный код бибилиотеки SimDataTypes состоит из двух файлов: SimDataTypes.h и SimDataTypes.cpp. В файле SimDataTypes.h содержат-

ся объявления структур данных и методов, используемых при разработке модулей расширения ПСС. В файле SimDataTypes.cpp содержатся определения методов, объявленных в файле SimDataTypes.h.

Рассмотрим эти структуры данных и методы.

10.1.1. Константы, определённые в библиотеке SimDataTypes

Ниже приводятся константы, определенные в библиотеке SimDataTypes. Значения предназначены для справки разработчику модулей, и их изменение не даст эффекта.

```
// Размер по умолчанию для буфера пакетов данных узла моделируемой сети
#define DEFAULT_PACKET_BUF_SIZE 50

// Размер по умолчанию для буфера статистики узла моделируемой сети
#define DEFAULT_MAX_PEER_STAT_SIZE 256

// Максимальное количество смежных узлов для узла моделируемой сети
//(если не переопределено в файле конфигурации ядра ПСС simulator.conf)
#define DEFAULT_MAX_NEIGHBOUVRS 50

// Максимальное количество циклов симуляции
//(если не переопределено в файле конфигурации ядра ПСС simulator.conf)
//Значение 0 означает «без ограничения»
#define DEFAULT_SIMULATION_TIME 0

//Число узлов в моделируемой сети по умолчанию
#define DEFAULT_PEER_COUNT 100
```

10.1.2. Структура настроек ядра ПСС SimSettings

Данная структура описывает настройки ядра ПСС, передаваемые всем моделирующим потокам от управляющего. Поскольку она не используется модулями расширения, ограничимся только её исходным объявлением без расшифровки назначения полей.

```
struct SimSettings
{
      unsigned int packetBufSize;
      unsigned int maxPeerStatSize;
      unsigned int maxNeighbours;
      unsigned int simulationTime;
      unsigned int peerCount;
      bool packetTeleportationIsAllowed;
      SimSettings();
};
```

10.1.3. Класс состояния ядра ПСС SimState

Класс SimState описывает текущее состояние ядра ПСС, в процессе моделирования объект этого класса существует в одном экземпляре в каждом моделирующем потоке и доступен модулям расширения через параметр simState метода DoWork интерфейса модуля расширения. Класс SimState объявлен в библиотеке SimDataTypes следующим образом:

```
class SimState
{
private:
      unsigned int lastStep;
      unsigned int simulationStep;
      unsigned int peerCount;
      unsigned int procNumber;
      unsigned int procCount;
      bool packetTeleportationIsAllowed;
      bool finished;
public:
      SimState(SimSettings simSettings, unsigned int procNumber,
unsigned int procCount);
      unsigned int GetSimulationStep();
      unsigned int GetLastStep();
      unsigned int GetPeerCount();
      unsigned int GetProcNumber();
      unsigned int GetProcCount();
      void NextStep();
      bool IsFinished();
      bool IsPacketTeleportationAllowed();
      void SetFinished();
};
```

Назначение публичных методов этого класса следующее:

- GetSimulationStep – возвращает номер текущего шага процесса моделирования. Используется в качестве таймера отсчёта текущего модельного времени. Ввиду того, что ПСС работает с дискретным временем, значение, возвращаемое GetSimulationStep является целым числом.
- GetLastStep – возвращает номер последнего шага моделирования, после которого процесс моделирования будет прерван[8]. Для установки этого параметра следует использовать параметр sim_max_simulation_steps конфигурационного файла ядра ПСС

[8] Следует заметить, что процесс моделирования может быть прерван раньше достижения этого шага по сигналу от одного из модулей расширения ПСС.

simulator.conf. Нулевое значение означает, что процесс моделирования не лимитируется количеством выполненных шагов, а должен быть прерван по каком-либо другому условию.

- GetPeerCount – возвращает общее количество узлов в моделируемой сети. В будущих версиях ПСС, которые будут разделять топологию сети между моделирующими потоками, возвращаемое значение по-прежнему будет представлять собой общее количество узлов (суммарное по всем моделирующим потокам).
- GetProcNumber – возвращает номер потока, который выполняет данный вызов модуля расширения[9]. Основное назначение данного метода – позволить модулям расширения в данной версии ПСС самостоятельно распределять вычислительную нагрузку между узлами вычислительной системы, на которой выполняется моделирование.
- GetProcCount – возвращает общее количество MPI-потоков, включая управляющий, выполняющих код ПСС. Используется совместно с GetProcNumber с целью распределения вычислительной нагрузки между узлами вычислительной системы, на которой выполняется моделирование.
- NextStep – увеличивает счётчик модельного времени. Предназначен для использования ядром ПСС. Вызов данного метода модулем расширения ПСС может привести к нарушению корректности отсчёта модельного времени и преждевременному завершению процесса моделирования.
- IsFinished – позволяет определить, выполнено ли условие завершения симуляции, а значит, является ли текущий цикл моделирования последним.
- IsPacketTeleportationAllowed – позволяет определить, разрешена ли отправка пакетов несмежным узлам модельной сети, а значит, имеет ли смысл модулю расширения выполнять такую отправку.
- SetFinished – позволяет модулю расширения указать ядру ПСС, что выполнено некоторое условие завершения процесс моделирования, и, следовательно, текущий шаг моделирования является последним.

---

[9] Следует помнить, что потоки нумеруются с нуля, а также тот факт, что нулевой поток – не моделирующий, а управляющий.

10.1.4. Структура пакета данных DataPacket

Данная структура описывает конструкцию пакета данных, пересылаемого по модельной сети. Она объявлена следующим образом:

```
struct DataPacket
{
      unsigned int creator;
      unsigned int finalDestination;
      unsigned int sender;
      unsigned int receiver;
      unsigned char ttl;
      unsigned int additionalDataSize;
      unsigned char * additionalData;
      DataPacket Clone();
};
```

Назначение полей этой структуры следующее:

- creator – идентификатор узла создавшего данный пакет. Является также идентификатором узла, впервые отправившего данный пакет. Для копий пакета, создаваемого промежуточными узлами модельной сети, выполняющими пересылку данного пакета, значение этого поля как правило сохраняется.
- finalDestination – узел назначения, являющийся получателем данного пакета. Этот узел может быть несмежным с текущим узлом, тогда доставка пакета возможна по некоторому маршруту с помощью модуля расширения, реализующего алгоритм (протокол) маршрутизации пакетов.
- sender – узел, отправивший данный пакет последним. При первой пересылке пакета совпадает с creator.
- receiver – очередной получатель пакета. Если отправка пакетов несмежным узлам модельной сети запрещена, то этот узел должен быть смежным узлом для текущего узла-отправителя пакета. В противном случае пакет будет уничтожен ядром ПСС.
- ttl – текущий остаток времени жизни пакета данных. Это значение показывает, какое максимальное количество пересылок ещё может претерпеть данный пакет до тех пор, пока не будет уничтожен ядром ПСС. Данное значение автоматически уменьшается на единицу ядром ПСС при каждой пересылке пакета. При попытке пересылки пакета с ttl = 0 пакет уничтожается. Основное назначение ttl – защита моделируемой сети от перегрузки в случае, когда моделируемые алгоритмы работают с ошибками (например алгоритм маршрутизации образует циклы).

- additionalDataSize – количество дополнительной информации пакета в байтах. Для повышения универсальности пакета данных в ПСС в его базовые поля вынесена только информация необходимая для функционирования ядра ПСС. Все остальные данные пакета должны храниться отдельно. При этом они могут иметь произвольную структуру и размер. Это позволяет выстраивать сложные иерархии протоколов различного уровня, каждый из которых обрабатывает различную информацию, содержащуюся в пакете.
- additionalData – указатель на область дополнительных данных пакета. По этому указателю размещаются дополнительные данные пакета. Эта область должна быть захвачена модулем расширения, создавшим пакет, перед размещением в ней данных.

10.1.5. Класс состояния узла моделируемой сети PeerState

Каждый объект класса PeerState содержит состояние одного из узлов моделируемой сети, а также методы, позволяющие манипулировать этим состоянием. Объекты класса PeerState доступны модулям расширения через параметр peerState метода DoWork интерфейса модуля расширения. Класс PeerState имеет следующее объявление:

```
class PeerState
{
private:
      bool fatalError;
      char * lastError;
      unsigned int peerId;
      bool isPowerOn;
      unsigned int inBufLevel;
      unsigned int maxInBufLevel;
      DataPacket * inBuffer;
      unsigned int outBufLevel;
      unsigned int maxOutBufLevel;
      DataPacket * outBuffer;
      unsigned int statBufLevel;
      unsigned int maxStatBufLevel;
      char * statBuffer;
      unsigned int neighboursCount;
      unsigned int maxNeighboursCount;
      unsigned int * neighbours;
public:
      PeerState();
      ~PeerState();
      void SetupPeer(unsigned int peerId, SimSettings simSettings);
      unsigned int GetPeerId();
```

```
        bool FatalErrorIsOccured();
        char * GetLastError();
        void ClearLastError();
        bool IsPowerOn();
        void SwitchOn();
        void SwitchOff();
        unsigned int GetInBufLevel();
        void ClearInBuf();
        void AddInPacket(DataPacket packet);
        void DelInPacket(unsigned int position);
        DataPacket * GetInPacket(unsigned int position);
        unsigned int GetOutBufLevel();
        void ClearOutBuf();
        void AddOutPacket(DataPacket packet);
        void DelOutPacket(unsigned int position);
        DataPacket * GetOutPacket(unsigned int position);
        char * GetStatistics();
        unsigned int GetStatBufLevel();
        void WriteStatistics(char * statistics);
        void ClearStatisitcs();
        unsigned int GetNeighboursCount();
        void ClearNeighbours();
        void AddNeighbour(unsigned int peerId);
        void DelNeighbour(unsigned int peerId);
        unsigned int GetNeighbourAt(unsigned int position);
};
```

Рассмотрим публичные методы класса PeerState:

- SetupPeer – данный метод предназначен для начальной установки информации об узле моделируемой сети в ходе инициализации ядра ПСС. Вызывается ядром ПСС, не предназначен для вызова модулями расширения ПСС.
- GetPeerId – возвращает идентификатор текущего узла моделируемой сети, уникальный в рамках моделируемой сети. Может быть использован для самых различны целей, в том числе для балансировки нагрузки между узлами вычислительной системы (совместно с соответствующими параметрами состояния ядра ПСС).
- FatalErrorIsOccured – позволяет выяснить, была ли в ходе последней операции фатальная ошибка. Предназначен для использования ядром ПСС.
- GetLastError – позволяет получить текст сообщения о последней ошибке, произошедшей в ходе работы модулей расширения ПСС. Используется ядром ПСС.

- ClearLastError – позволяет очистить текст сообщения о последней ошибке, произошедшей в ходе работы модулей расширения ПСС. Используется ядром ПСС.
- IsPowerOn – позволяет определить включен или выключен текущий узел моделируемой сети. Выключенные узлы не получают и не отправляют пакеты данных (даже если модуль расширения ПСС разместит пакеты в буфере для сходящих пакетов).
- SwitchOn – позволяет включить текущий узел моделируемой сети. Если узел уже включен, вызов данного метода не производит эффекта.
- SwitchOff – позволяет выключить текущий узел моделируемой сети. Если узел уже выключен, вызов данного метода не производит эффекта.
- GetInBufLevel – позволяет определить количество пакетов данных в буфере для входящих пакетов текущего узла моделируемой сети. Данный метод должен использоваться любым модулем расширения ПСС, работающим с буфером для входящих пакетов данных, так как попытка получить пакет из позиции буфера, которая на текущий момент не используется, трактуется ядром ПСС как фатальная ошибка, что приводит к немедленному аварийному завершению процесса моделирования.
- ClearInBuf – позволяет очистить буфер для входящих пакетов текущего узла моделируемой сети. При этом уровень буфера входящих пакетов данных этого узла становится равен нулю.
- AddInPacket – добавляет пакет в буфер для входящих пакетов данных текущего узла моделируемой сети. В случае если буфер уже заполнен, добавление не происходит, а ядро ПСС констатирует возникновение нефатальной ошибки, сообщение о которой пересылается управляющему потоку для записи его в файл журнала текущего сеанса моделирования ПСС.
- DelInPacket – удаляет пакет данных из указанной позиции буфера входящих пакетов текущего узла моделируемой сети. Если указанная позиция буфера ещё не используется, то ядром ПС генерируется фатальная ошибка, приводящая к

преждевременному аварийному завершению процесса моделирования, поэтому модуль расширения ПСС должен контролировать уровень буфера входящих пакетов текущего узла моделируемой сети при удалении из него пакетов данных. Кроме того, разработчику модуля расширения ПСС следует иметь в виду, что в текущей версии ядра ПСС при удалении пакетов порядок расположения оставшихся в буфере пакетов изменяется.

- GetInPacket – получение пакета из указанной позиции буфера входящих пакетов. Если указанная позиция буфера ещё не используется, то ядром ПС генерируется фатальная ошибка, приводящая к преждевременному аварийному завершению процесса моделирования, поэтому модуль расширения ПСС должен контролировать уровень буфера входящих пакетов текущего узла моделируемой сети при запросе из него пакетов данных.
- GetOutBufLevel – позволяет определить количество пакетов данных в буфере для исходящих пакетов текущего узла моделируемой сети. Данный метод должен использоваться любым модулем расширения ПСС, работающим с буфером для исходящих пакетов данных, так как попытка получить пакет из позиции буфера, которая на текущий момент не используется, трактуется ядром ПСС как фатальная ошибка, что приводит к немедленному аварийному завершению процесса моделирования.
- ClearOutBuf – позволяет очистить буфер для исходящих пакетов текущего узла моделируемой сети. При этом уровень буфера исходящих пакетов данных этого узла становится равен нулю.
- AddOutPacket – добавляет пакет в буфер для исходящих пакетов данных текущего узла моделируемой сети. В случае если буфер уже заполнен, добавление не происходит, а ядро ПСС констатирует возникновение нефатальной ошибки, сообщение о которой пересылается управляющему потоку для записи его в файл журнала текущего сеанса моделирования ПСС.

- DelOutPacket – удаляет пакет данных из указанной позиции буфера исходящих пакетов текущего узла моделируемой сети. Если указанная позиция буфера ещё не используется, то ядром ПСС с генерируется фатальная ошибка, приводящая к преждевременному аварийному завершению процесса моделирования, поэтому модуль расширения ПСС должен контролировать уровень буфера исходящих пакетов текущего узла моделируемой сети при удалении из него пакетов данных. Кроме того, разработчику модуля расширения ПСС следует иметь в виду, что в текущей версии ядра ПСС при удалении пакетов порядок расположения оставшихся в буфере пакетов изменяется.
- GetOutPacket – получение пакета из указанной позиции буфера исходящих пакетов. Если указанная позиция буфера ещё не используется, то ядром ПСС с генерируется фатальная ошибка, приводящая к преждевременному аварийному завершению процесса моделирования, поэтому модуль расширения ПСС должен контролировать уровень буфера исходящих пакетов текущего узла моделируемой сети при запросе из него пакетов данных.
- GetStatistics – получение статистики работы данного узла моделируемой сети, накопленной в ходе текущего шага моделирования[10]. Статистика генерируется модулями расширения ПСС в ходе обработки данного узла ПСС.
- GetStatBufLevel – получение количество байт, используемых в настоящий момент в буфере статистики текущего узла моделируемой сети. Может быть использовано для определения того, выполнял ли на данном шаге моделирования какой-либо модуль расширения ПСС запись статистики в буфер текущего узла моделируемой сети. Преимущественно используется ядром ПСС.

---

[10] Следует иметь в виду, что в конце каждого шага моделирования после пересылки статистики работы узла управляющему потоку ПСС для сохранения её в файл журнала текущего сеанса моделирования ПСС данный буфер очищается.

- WriteStatistics – добавление статистики в буфер статистики текущего узла моделируемой сети. В случае, если добавляемое сообщение не помещается в остаточный объём буфера статистики, ядром ПСС фиксируется соответствующая нефатальная ошибка, а добавление статистики не производится.
- ClearStatisitcs – очистка буфера статистики текущего узла моделируемой сети. Обычно используется ядром ПСС, но может быть использовано и модулем расширения ПСС, например, в ситуации, когда требуется произвести запись в буфер статистики какой-либо важной информации.
- GetNeighboursCount – позволяет получить для текущего узла моделируемой сети количество узлов, смежных с ним. Для узлов, не имеющих смежных, возвращается 0.
- ClearNeighbours – позволяет очистить список смежных узлов для текущего узла моделируемой сети. Следует иметь в виду, что в текущей реализации ядра ПСС это не приведёт к удалению информации о смежности с текущим узлом моделируемой сети для тех узлов моделируемой сети, с которыми был связан текущий узел моделируемой сети. То есть, если между узлами была двунаправленная связь, она станет однонаправленной. Принятие при необходимости мер по удалению обратных связей возлагается на разработчика модуля расширения ПСС.
- AddNeighbour – добавление смежного узла для текущего узла моделируемой сети[11]. Следует иметь в виду, что создаваемая связь является однонаправленной, то есть только данный узел оказывается смежным с указанным, но не наоборот. Для создания двунаправленной связи, в случае если обратная связь отсут-

[11] Так как объём буфера для связей узла моделируемой сети ограничен, при попытке добавления новой связи может произойти его переполнение. В этом случае добавление не производится, а ядро ПСС констатирует возникновение нефатальной ошибки. Также нефатальная ошибка происходит при попытке добавления узла в смежные к самому себе и при попытке повторного добавления в смежные узла, который уже является смежным с данным узлом.

ствует, следует дополнительно создать и её, вызвав аналогичный метод у узла моделируемой сети, связываемого с данным.

- DelNeighbour – удаление узла из списка смежных с текущим узлом. Для полного удаления связи между узлами, в случае, если она была двунаправленной, следует дополнительно удалить вызовом аналогичного метода текущий узел из списка смежных с удаляемым. В текущей версии ядра ПСС эта задача возлагается на разработчика модуля расширения ПСС. Попытка удаления из списка смежных узлов моделируемой сети узла, не присутствующего в этом списке, не производит никакого эффекта.
- GetNeighbourAt – получение идентификатора одного из узлов моделируемой сети, смежного с текущим узлом моделируемой сети. Следует иметь в виду, что если запрошенная позиция больше или равна количеству смежных узлов, ядро ПСС констатирует возникновение фатальной ошибки и немедленно производит аварийное завершение процесса моделирования. Поэтому разработчик модуля расширения, манипулирующего со списком смежных узлов для данного узла моделируемой сети, должен сначала получить количество таких узлов посредством вызова метода GetNeighboursCount и действовать в соответствии с полученной информацией.

10.1.6. Класс интерфейса модуля расширения ПСС IExtensionModule

Данный класс наследуется всеми модулями расширения ПСС. Назначение его методов описано в разделе, посвящённом интерфейсу модуля расширения ПСС, а объявление имеет следующий вид:

```
class IExtensionModule
{
public:
      virtual void DoWork(PeerState * peerState, SimState * simState) = 0;
      virtual void Shutdown(int * bufSize, char ** buf) = 0;
};
```

10.2. Средства повышения производительности в пределах вычислительного узла, реализованные в архитектуре ПСС

Современные процессоры имеют, как правило, несколько вычислительных ядер, однако технология MPI предназначена для использования в первую очередь в системах с распределённой памятью. В результате, в типичном случае применения ПСС все ядра процессоров вычислительных узлов кроме первого оказываются незадействованными.

В текущей версии имеется частичное использование двух ядер на вычислительном узле, за счёт наличия в каждом вычислительном потоке ПСС[12] двух потоков, один из которых обеспечивает работу сервера обмена сообщениями между моделирующими потоками, а второй – собственно моделирование.

Реализуется это следующим образом:

- При запуске ПСС каждый его MPI-процесс (главный поток) запускает ещё один поток – сервер сообщений.
- Сервер сообщений ожидает сообщений от других MPI-процессов ПСС, реализующих другие моделирующие или управляющий процессы, и помещает их в динамический вектор, откуда они могут быть в дальнейшем извлечены.
- При необходимости проверки наличия входящих сообщений от других процессов главный поток процесса обращается к вектору входящих сообщений с целью проверки его непустоты и извлечения, в случае если он не пуст, очередного сообщения.
- При этом для обеспечения корректности работы с памятью используются средства синхронизации потоков ОС Linux, реализуемые посредством библиотеки pthread.

---

[12] С технической точки зрения правильнее вычислительные и управляющий потоки ПСС называть MPI-процессами или просто процессами, хотя в данных указаниях всюду используется термин «MPI-поток» или просто «моделирующий поток». Однако в данном пункте отчёта во избежание коллизии терминов понятия «поток» и «процесс» употребляются в своём истинном значении.

11. ПРИМЕР МОДУЛЯ РАСШИРЕНИЯ ПСС

В данном разделе рассматривается пример простейшего модуля расширения ПСС, реализующего лавинную рассылку пакетов. Принцип лавинной рассылки заключается в том, что при поступлении в узел пакета, его копии рассылаются всем узлам, смежным с данным узлом, за исключением узла, приславшего пакет. Смежные узлы поступают аналогичным образом, и процесс лавинообразно продолжается до истечения TTL (остаточного времени жизни) копий пакета.

Для реализации данного модуля расширения ПСС необходимо создать динамическую библиотеку (например библиотеку PacketReplicator, то есть имеющую имя файла libPacketReplicator.so), включающую описанную ранее статическую библиотеку SimDataTypes. Типовая схема проведения эксперимента дана в Прил. 4.

Исходный код рассматриваемого примера может состоять из трёх файлов, рассмотренных ниже.

11.1. Главный файл модуля расширения ПСС Main.cpp

Файл Main.cpp предназначен для создания объекта модуля расширения ПСС при инициализации модуля расширения в процессе инициализации ПСС. Он состоит из одного метода Init, описание назначения и параметров которого приведено в описании интерфейса модуля расширения ПСС. Код такого метода может быть следующим:

```
#include "PacketReplicator.h"
#include <stdio.h>

extern "C" IExtensionModule * Init(char * config)
{
	return new PacketReplicator();
}
```

Принцип его работы простой: при вызове метода Init создать объект модуля расширения и вернуть его в точку вызова. Для предотвращения искажения имён функций, экспортируемых из динамической библиотеки, в определении метода Init использован модификатор extern "C".

11.2. Заголовочный файл модуля расширения ПСС (PacketReplicator.h)

Данный файл содержит объявление класса модуля расширения, который обязательно является наследником интерфейса модуля расширения IExtensionModule, описанного выше в описании статической библиотеки SymDataTypes. Класс модуля расширения реализует методы, описанные выше в разделе «Интерфейс модуля расширения ПСС».

Пример содержимого заголовочного файла модуля расширения ПСС:

```
#ifndef PACKETREPLICATOR_H_
#define PACKETREPLICATOR_H_

#include <SimDataTypes.h>

class PacketReplicator : public IExtensionModule
{
private:
      unsigned long counter;
public:
      PacketReplicator();
      void DoWork(PeerState * peerState, SimState * simState);
      void Shutdown(int * bufSize, char ** buf);
};


#endif /* PACKETREPLICATOR_H_ */
```

Данное определение класса содержит:

- объявление конструктора без параметров, вызываемого в методе Init;
- объявление метода DoWork, реализующего функцию данного модуля расширения ПСС – репликацию пакетов;
- объявление метода Shutdown, реализующего возврат финальной статистики работы данного модуля расширения ПСС;
- объявление закрытого для доступа извне поля counter для реализации внутренних нужд данного модуля расширения ПСС – в данном случае для подсчёта пакетов, сгенерированных модулем расширения.

11.3. Файл определения модуля расширения ПСС (PacketReplicator.cpp)

Данный файл определяет методы, объявленные в заголовочном файле данного модуля расширения ПСС. Он может быть таким:

```
#include "PacketReplicator.h"
#include <stdio.h>
#include <string.h>

PacketReplicator::PacketReplicator()
{
	counter = 0;
}

void PacketReplicator::DoWork(PeerState * peerState, SimState *
simState)
{
	for(unsigned int i=0; i < peerState->GetInBufLevel(); i++)
	{
		DataPacket * dataPacket =  peerState->GetInPacket(i);
		if(dataPacket == NULL || dataPacket->ttl == 0)
		{
			continue;
		}
		for(unsigned int j=0; j<peerState->GetNeighboursCount(); j++)
		{
			if(peerState->GetNeighbourAt(j) != dataPacket->sender)
			{
				DataPacket newPacket = dataPacket->Clone();
				newPacket.receiver = peerState->GetNeighbourAt(j);
				peerState->AddOutPacket(newPacket);
				counter++;
			}
		}
	}
}

void PacketReplicator::Shutdown(int * bufSize, char ** buf)
{
	char sbuf[256];
	sprintf(sbuf, "Produced %lu data packets", counter);
	*bufSize = strlen(sbuf) + 1;
	*buf = new char[*bufSize];
	strcpy(*buf, sbuf);
}
```

Данный листинг содержит:

- Определение конструктора, инициализирующего нулем счётчик сгенерированных пакетов.
- Определение метода DoWork, анализирующего пакеты в буфере входящих пакетов рассматриваемого узла и создающего копий всех пакетов имеющих ненулевое остаточное время жизни в количестве на один меньше количества соседей, помещая затем созданные копии в буфер исходящий пакетов текущего узла с указанием соответствующих адресов узлов-получателей пакетов. При этом инкрементируется счётчик созданных пакетов.
- Определение метода финализации модуля расширения (Shutdown), возвращающего в качестве финальной статистики количество созданных пакетов с необходимым комментарием.

ПРИЛОЖЕНИЯ

Приложение 1

Перечень лабораторных работ

Лабораторные работы подобраны таким образом, чтобы в результате их выполнения получалась готовая функционирующая модель сети, с которой в дальнейшем можно проводить различные эксперименты. После выполнения каждой из работ (с первой по четвертую) модуль расширения ПСС, созданный в ходе её выполнения, должен быть протестирован на работоспособность.

Лабораторная работа 1. Разработка модуля генератора топологии

В ходе выполнения данной лабораторной работы должен быть создан модуль генерации топологии, который производит соединение узлов модели между собой путём заполнения списков смежных узлов. Топология предполагается статической, поэтому модуль должен быть организован таким образом, чтобы он выполнял свою работу для каждого узла только один раз (на первом цикле работы симулятора). Необходимые параметры работы модуля, задаваемые в его конфигурационном файле, определить самостоятельно в соответствии с вариантом задания.

Вариант №	Задание
1	Генерация топологии сети, в которой каждый узел соединён ровно с N другими однонаправленными связями.
2	Генерация топологии сети, в которой каждый узел соединён с другими произвольным количеством двунаправленных связей.
3	Генерация сети, в которой число узлов, смежных с данным, распределено равномерно в диапазоне от N до M.
4	Генерация полносвязной сети.
5	Генерация сети с топологией «кольцо» (все связи двунаправленные).
6	Генерация сети с топологией «шина» (все связи двунаправленные).
7	Генерация сети с топологией «звезда» (все связи двунаправленные).

Лабораторная работа 2. Разработка модуля генератора трафика

В ходе выполнения данной лабораторной работы должен быть создан модуль генерации пакетов данных, который производит создание пакетов данных в узлах сети. При этом необходимо предусмотреть, чтобы в каждом из N расчётных процессов пакеты генерировались на соответствующей 1/N узлов, которые могут по условиям эксперимента быть источниками трафика (это обеспечит правильное распределения нагрузки между расчётными процессами ПСС). Необходимые параметры работы модуля, задаваемые в его конфигурационном файле, определить самостоятельно в соответствии с вариантом задания.

Вариант №	Задание
1	Генерация на первом шаге моделирования одного пакета на каждом узле сети.
2	Генерация на каждом шаге моделирования в два раза меньше пакетов (суммарно по всей сети), чем на предыдущем.
3	Генерация пакета в узле, если на предыдущем шаге моделирования пакет в нём не генерировался.
4	Генерация на каждом шаге моделирования пакета только в одном узле сети (номер узла выбрать случайно).
5	Генерация на каждом шаге пакета на узле, идентификатор которого соответствует номеру шага моделирования.
6	Генерация на первом шаге моделирования пакетов на узлах с чётными идентификаторами, а на втором – с нечётными.
7	Генерация на первом шаге суммарно N пакетов на N различных узлов со случайно выбранными идентификаторами.

Лабораторная работа 3. Разработка модуля протокола маршрутизации

В ходе выполнения данной лабораторной работы должен быть создан модуль протокола маршрутизации пакетов данных, который производит определение пути транспортировки созданных пакетов. Необходимые параметры работы модуля, задаваемые в его конфигурационном файле, определить самостоятельно в соответствии с вариантом задания. Во всех вариантах предусмотреть параметр TTL (время жизни пакета), задающий количество пересылок пакета, после которых он должен уничтожаться.

Вариант №	Задание
1	Эхо-запрос: лавинная рассылка пакета с идентификатором искомого узла и с накоплением пути в содержимом пакета; затем узел, получив пакет со своим идентификатором, посылает его обратно по пути, по которому он дошёл.
2	Лавинная рассылка с исключением повторных рассылок пакета, пришедшего другим маршрутом.
3	Маршрутизация по заданному маршруту (маршрут задаётся в виде последовательности номеров узлов).
4	Ветвящееся случайное блуждание.
5	Случайное блуждание.
6	Пересылка пакетов узлам с большим номером.
7	Лавинная рассылка без исключения повторных рассылок пакета, пришедшего другим маршрутом.

Лабораторная работа 4. Разработка модуля сборщика статистики

В ходе выполнения данной лабораторной работы должен быть создан модуль сбора статистики работы модели, который собирает и сохраняет статистику состояния модели в ходе моделирования. Собираемая статистика должна направляться в журнал сеанса моделирования через буферы статистики узлов или метод Shutdown модуля сборщика статистики (в зависимости от того, должен ли быть получен один результат на весь процесс моделирования или отдельный результат на каждом шаге моделирования). Варианты заданий, определяющие тип собираемых данных приведены в ниже:

Вариант №	Задание
1	Измерение длительности выполнения шага моделирования.
2	Определение количества пересылок пакетов на текущем шаге моделирования.
3	Определение средней длительности выполнения шага моделирования.
4	Определение среднего количества пересылок пакетов на шаге моделирования.
5	Определение длительности моделирования.
6	Определение суммарного количества пересылок пакетов за время моделирования.
7	Определение среднего количества пакетов, приходящих на шаге моделирования.

Приложение 2

Типовой набор модулей расширения

Для выполнения эксперимента с использованием ПСС требуется предварительно разработать набор модулей расширения, реализующих логику эксперимента.

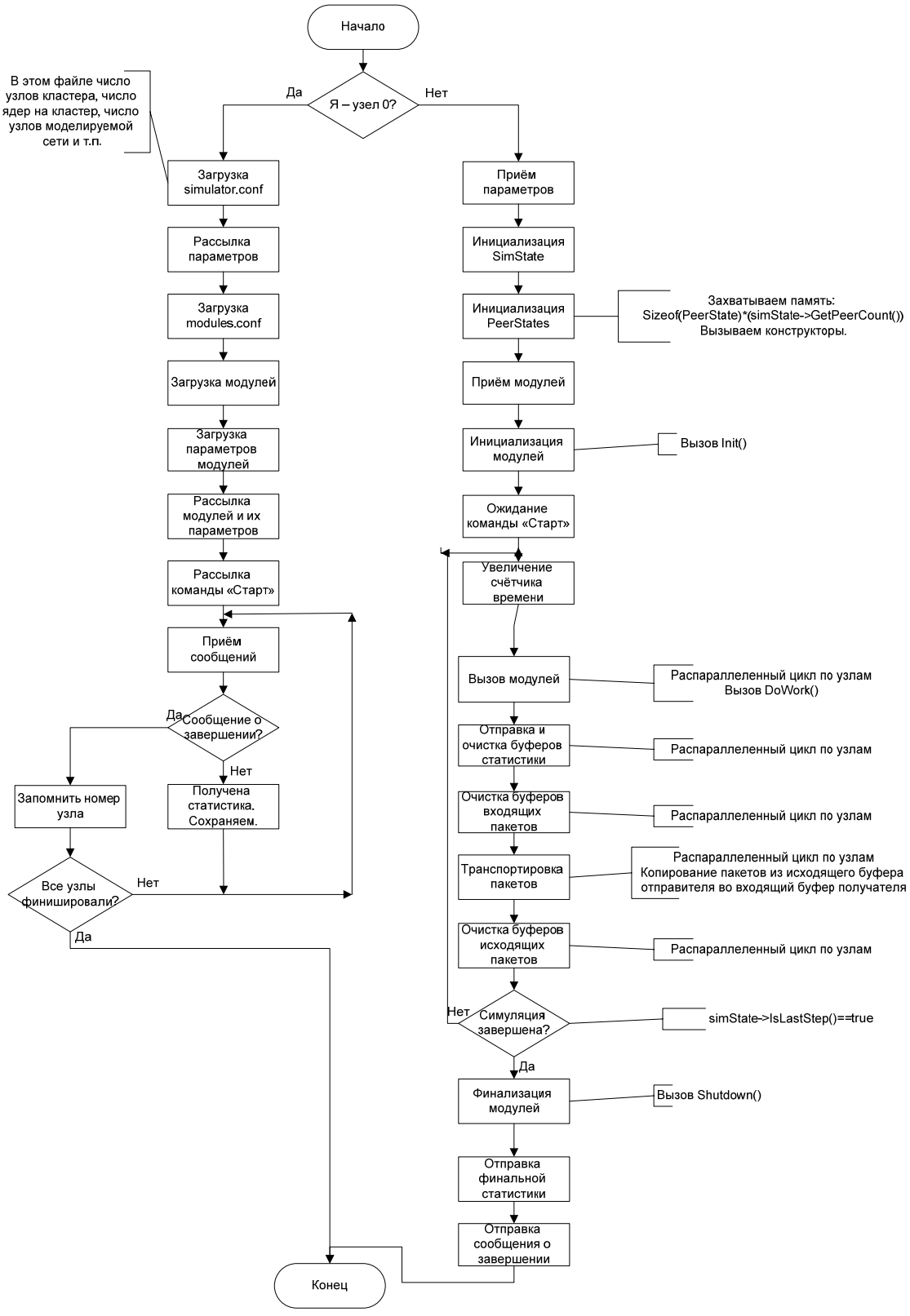
В простейшем случае это будет следующий набор:

1. Генератор топологии – должен создать топологию сети и, в случае моделирования сети с переменной топологией, изменять её с течением модельного времени. Этот модуль не требуется в случае полносвязной топологии, которую можно задать с помощью специального параметра конфигурации ядра ПСС.
2. Генератор трафика – для создания пакетов данных, которые будут передаваться по моделируемой сети. Этот модуль не нужен, если предметом исследования является сама топология.
3. Протокол маршрутизации, который будет обеспечивать определение пути транспортировки созданных пакетов. Этот модуль не нужен в случае полносвязной топологии, так как доставка пакетов смежным узлам реализована в ядре ПСС.
4. Сборщик статистики. Будет собирать интересующую нас статистику состояния модели в ходе моделирования. Этот модуль не нужен, если данная функция реализована в других модулях расширения.

Для обеспечения правильного распределения нагрузки между расчётными процессами ПСС проще всего реализовать модуль генератора трафика (если он предусмотрен) таким образом, чтобы в каждом из N расчётных процессов пакеты генерировались на соответствующей доли узлов, которые могут по условиям эксперимента быть источниками трафика.

Приложение 3

Блок-схема алгоритма работы ПСС



Приложение 4

Типовая схема проведения эксперимента

Для выполнения машинного эксперимента с использованием ПСС требуется выполнить следующую последовательность действий:

1. Настроить файлы конфигурации ПСС, указав в частности используемый набор модулей расширения.
2. Настроить файлы конфигурации модулей расширения.
3. Запустить процесс моделирования на необходимом количестве узлов используемой вычислительной системы с помощью имеющихся в наличии средств управления заданиями.
4. По завершении моделирования проанализировать его результаты путём изучения файла журнала сеанса работы ПСС, в который помещается вся статистика состояния модели, собранная в ходе моделирования.

Библиографический список

1. Шамин, П.Ю. Параллельный сетевой симулятор: концепция использования и перспективы развития / П.Ю. Шамин, А.С. Алексанян, В.В. Прокошев // Научно-технические ведомости Санкт-Петербургского государственного политехнического университета. – 2009. – № 3. – С. 18 – 24. – ISSN 1994-2354.
2. Шамин, П.Ю. Разработка системы моделирования сетей большой размерности на базе параллельного сетевого симулятора ПСС / П.Ю. Шамин, А.С. Алексанян, В.В. Прокошев // Телематика-2009: тр. XVI Всерос. науч.-метод. конф. – СПб., 2009. – Т. 2. – С. 379 – 381. – ISBN 978-5-7577-0337-4.
3. Алексанян, А.С. Параллельный сетевой симулятор: концепция использования и перспективы развития / А.С. Алексанян, С.М. Аракелян, В.В. Прокошев, П.Ю. Шамин // Информатизация образования и науки. – 2009. – № 4. – С. 88 – 101. – ISSN 2073-7572.

Оглавление

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к лабораторным работам по курсу «Компьютерное моделирование»

Составитель
ШАМИН Павел Юрьевич

Ответственный за выпуск – зав. кафедрой профессор С.М. Аракелян

Подписано в печать 17.03.10.
Формат 60x84/16. Усл. печ. л. 2,79. Тираж 70 экз.
Заказ
Издательство
Владимирского государственного университета.
600000, Владимир, ул. Горького, 87.