

Содержание

Введение	7
1 Системы мониторинга.....	8
1.1 Введение.....	8
1.2 Понятие систем мониторинга	8
1.3 Подсистемы мониторинга	9
1.3.1 Сбор данных.....	10
1.3.2 Хранение данных	11
1.3.3 Анализ данных.....	11
1.3.4 Отчетность	12
1.3.5 Оповещения	12
1.3.6 Диспетчеризация.....	13
1.4 Классификация систем мониторинга	13
1.5 Требования к системам мониторинга.....	15
1.6 Проблемы эксплуатации систем мониторинга	16
1.7 Выводы.....	17
2 Модель распределенной системы мониторинга	19
2.1 Общие положения	19
2.1 Базовая теоритическая модель.....	19
2.2 Модуль мониторинга.....	20
2.3 Система исполнения.....	21
2.4 Код каркаса	22
2.5 Прикладной интерфейс программирования.....	23
2.6 Состояние распределенной системы мониторинга	23

3 Реализация системы	26
3.1 Служба мониторинга.....	26
3.1.1 Структура службы мониторинга.....	26
3.1.2 Выбор средств реализации	27
3.1.3 Ядро системы	32
3.1.4 Транспортная подсистема	56
3.1.5 Подсистема исполнения	61
3.2 Менеджер модулей.....	64
3.2.1 Общее описание.....	64
3.2.2 Выбор средств реализации	65
3.2.3 Уникальный идентификатор модуля	65
3.3 Прикладной интерфейс программирования	66
3.3.1 Общее описание.....	66
3.3.2 Выбор средств реализации	66
3.4 Панель управления	67
3.4.1 Общее описание.....	67
3.4.2 Архитектура панели управления	68
3.4.3 Модель.....	69
3.4.4 Контроллер.....	74
3.4.5 Представление	75
3.5 Описание организации совместной работы	79
3.5.1 Skype (skype.com)	80
3.5.2 GoogleMail (gmail.com).....	80
3.5.2 Хостинг проектов Google (goolecode.com).....	80
4 Организационно-экономический раздел.....	81

4.1 Расчет затрат на этапе проектирования.....	81
4.2 Выбор базы сравнения	86
4.3 Сравнительный анализ затрат в ходе эксплуатации программного продукта и аналога	88
4.4 Расчет экономии от увеличения производительности труда пользователя	91
4.5 Ожидаемый экономический эффект и срок окупаемости капитальных затрат.....	92
5 Охрана труда и окружающей среды.....	95
5.1 Аттестация рабочего места программиста.....	95
5.1.1 Шум	96
5.1.2 Искусственная освещенность	97
5.1.3 Неионизирующие электромагнитные поля и излучения.....	99
5.1.4 Напряженность трудового процесса.....	99
5.1.5 Взрывопожаробезопасность.....	101
5.1.6 Электробезопасность.....	104
5.1.7 Травмобезопасность	105
5.2 Обзор альтернативных программных решений с точки зрения повышения производительности труда	107
5.2.1 Обзор системы Zabbix	107
5.2.1.1 Общие сведения	107
5.2.1.2 Описание основных функций	109
5.2.1.3 Удобство использования	113
5.2.1.4 Повышение производительности управление учетными записями.....	116
5.2.1.5 Поддерживаемые платформы	117

5.2.2 Превосходство над аналогами	118
5.3 Выводы.....	118
Заключение	120
Список использованных источников	122
Приложение А Задание на дипломное проектирование	125
Приложение Б Руководство администратора	127
Приложение В Код программы	135

Введение

Быстрорастущий уровень современной компьютеризации общества сопровождается появлением нового класса программных инструментов – систем мониторинга. Основная задача подобных решений - систематический анализ и интерпретация протекающих в гетерогенной среде процессов. Полученные в результате мониторинга данные могут быть использованы как для улучшения процесса принятия решений, так и для выявления узких мест исследуемой системы.

Настоящая работа представляет собой исследование современных решений в области мониторинга, оценку их эффективности и применимости согласно выдвинутой модели требований, а также выводы о необходимости появления нового класса инструментов мониторинга, ввиду неготовности существующих решений удовлетворять ранее выдвинутым требованиям.

Кроме того, в работе детально представлена предлагаемая авторами модель распределенной системы мониторинга гетерогенной среды, лишенная недостатков классических клиент-серверных систем. В основе предлагаемой архитектуры лежат механизмы разработки и исполнения дополнительных модулей, а также возможности и свойства отказоустойчивых распределенных систем.

Наконец, в работе подробно описана спроектированная архитектура каркаса мультиплатформенной распределенной системы мониторинга и рассмотренна её реализация с точки зрения современных технологий программирования распределенных систем – высокоуровневых языков и библиотек среднего слоя.

Итогом работы является каркас распределенной системы мониторинга, который представляет собой программный комплекс, состоящий из службы мониторинга, менеджера модулей, интерфейса программирования модулей и панели управления.

1 Системы мониторинга

1.1 Введение

Понятие систем мониторинга впервые было рассмотрено в начале 80-х годов XXI века, в период становления и популяризации сетевых операционных систем. В это время, системы представляли собой унифицированные инструменты сетевого администратора, необходимые и пригодные для обнаружения отказа оборудования или построения примитивной статистики использования ресурсов.

В настоящее время, системы мониторинга являются одним из самых насыщенных классов программных систем [link], предоставляющих пользователю широчайших набор возможностей, ориентированных на различные задачи и среды эксплуатации.

Системы мониторинга, как класс программных систем за почти полувековую историю эволюционировали из примитивных инструментов администратора до универсальных коробочных решений промышленного уровня.

Несмотря на достаточную насыщенность рынка, новые инструменты мониторинга, совмещающие в себе современные подходы и методологии программирования, продолжают появляться, определяя новые вектора развития класса систем в целом.

1.2 Понятие систем мониторинга

Мониторинг — систематический сбор и анализ информации о состоянии некоторого вычислительного узла или информационного процесса, организованный с определенной целью. Целями мониторинга могут быть: формализация и улучшение процесса принятия решений; детальный анализ и исследование системы на предмет поиска узких, высоконагруженных мест; сокращение времени простоя системы в случае выхода из строя ее основных компонентов и т.п.

Базовая теоретическая модель описывается с помощью понятий вычислительного узла, сервера и агента мониторинга (рисунок 1.1).

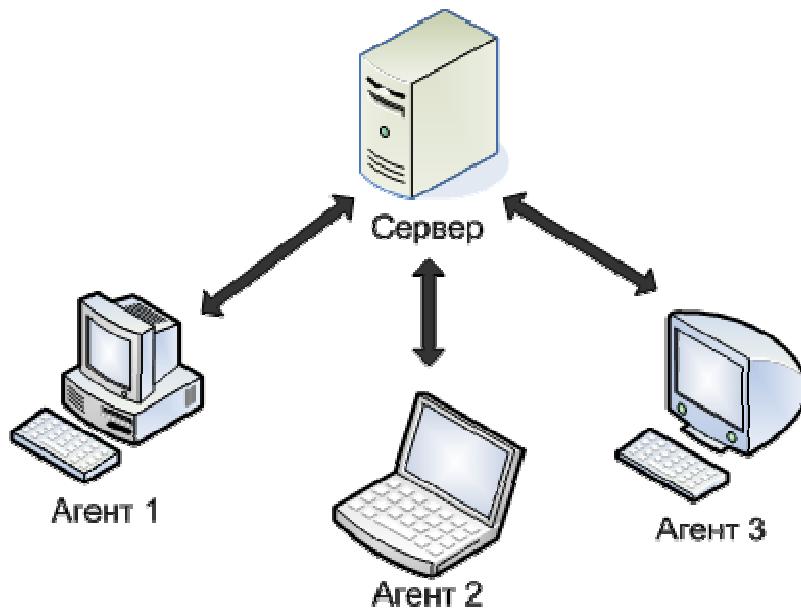


Рисунок 1.1 – Базовая модель

Под *вычислительным узлом* гетерогенной среды здесь и далее будем понимать программно-аппаратное устройство, в память которого может быть загружен и затем исполнен код какой-либо сущности мониторинга.

Агент, запущенный на определенном узле, представляется активной сущностью, непрерывно наблюдающей за его состоянием и передающей серверу сообщения об изменении этого состояния.

Сервер мониторинга — пассивная сущность, предоставляющая агентам ресурсы для приема сообщений, их последующей обработки и хранения, а также реализующий механизмы отчетности и реагирования на нештатные ситуации.

1.3 Подсистемы мониторинга

Функционирование любой системы мониторинга можно представить в виде набора взаимосвязанных повторяющихся действий, среди которых наиболее концептуальными являются сбор, хранение и анализ данных, а также отчетность и оповещение. Тогда обобщенная архитектура системы мониторинга будет

выглядеть как композиция отдельных подсистем, ответственных за каждое из вышеперечисленных действий (рисунок 1.2).



Рисунок 1.2 – Подсистемы мониторинга

1.3.1 Сбор данных

Известно несколько подходов к сбору данных системой мониторинга с удаленных вычислительных узлов гетерогенной среды, среди которых можно выделить два наиболее распространенных: двусторонний и односторонний. Двусторонний метод сбора данных представляет собой исследование и анализ реакции удаленной системы на определенный набор внешних воздействий. При одностороннем методе, исследуемая система сама предоставляет исследователю необходимые данные путем пересылки оповещений об изменении своего внутреннего состояния.

С технической точки зрения двусторонний метод является более предпочтительным при построении систем мониторинга, т.к. требует меньших затрат на реализацию и максимально использует возможности исследуемой операционной среды. В свою очередь, односторонний метод предоставляет больше возможностей по наращиванию функционала конечной системы.

На практике чаще всего используется комбинированный метод сбора данных, который объединяет в себе особенности двух подходов – одностороннего и двустороннего. Комбинированный метод сбора данных характеризуется использованием там, где это возможно, двустороннего подхода и одностороннего во всех остальных случаях.

Подсистему сбора данных разделяют между собой агенты и сервер мониторинга. Все остальные рассмотренные подсистемы относятся только к серверу мониторинга.

1.3.2 Хранение данных

Хранение данных, полученных в результате процессов мониторинга, может быть организовано как с использованием средств баз данных, так и на базе простых плоских файлов. Существуют также «задаче-ориентированные» варианты хранилищ, например распределенные высоконагруженные кеши, облачные нереляционные базы данных и т.п. Таким образом, варианты хранения данных можно охарактеризовать как централизованные и децентрализованные.

Очевидно, что децентрализованные или распределенные варианты хранения обладают большей отказоустойчивостью взамен сложности реализации и сопровождения.

Можно сформулировать минимальный, с точки зрения авторов, список требований к подсистеме хранения данных (централизованной или децентрализованной) системы мониторинга:

- а) целостность, доступность и безопасность данных;
- б) производительность и эффективность примитивных операций ввода вывода;
- в) легкость внедрения и сопровождения.

1.3.3 Анализ данных

Анализ, оценка и принятие решений могут происходить непосредственно в реальном времени, как реакция на многократное возникновение нештатной ситуации.

Подсистема анализа данных реализует механизмы и примитивы проверки полученных в результате мониторинга данных для выявления текущего состояния исследуемой системы. Подсистему анализа можно представить в виде композиции следующих логических компонент:

- а) цепочка правил или предикатов проверки данных;
- б) механизмы применения правил;
- в) механизмы генерации исключительных ситуаций.

В результате применения цепочки правил к каждым полученным данным, подсистема анализа выявляет текущее состояние удаленной системы, которое может характеризовать работу исследуемой системы как штатную или нештатную.

При определении нештатной работы удаленной системы, подсистемой анализа генерируются исключительные ситуации, обрабатываемые подсистемой оповещений или диспетчеризации.

Анализ данных мониторинга позволяет выявить тенденции нежелательных ситуаций.

1.3.4 Отчетность

Отчеты позволяют визуализировать и кластеризовать данные мониторинга в удобочитаемой форме, пригодной для анализа и просмотра пользователем. Подсистема генерации отчетов позволяет представить данные в виде информации, которую можно распечатать или сохранить в одном из поддерживаемых электронных форматах.

По способу визуализации можно выделить текстовые и графические отчеты. К текстовым отчетам относятся непосредственно текст, списки, таблицы. Этот вид отчетов характеризуется большим объемом информации при компактном отображении, что делает его удобным для хранения. К графическим относятся графики, графы, схемы. Основным достоинством этого вида отчетов является наглядность, быстрота восприятия. Эти отчеты эффективны, когда необходимо динамично и понятно отобразить данные.

1.3.5 Оповещения

Подсистема оповещений реализует набор инструментов и механизмов оповещения заинтересованных клиентов о возникновении исключительных

ситуаций в исследуемой системе. Под исключительными понимаются ситуации, приводящие к сбою в работоспособности или отказу аппаратной или программной части системы.

В некотором смысле, можно представлять систему оповещения как набор обработчиков исключительных ситуаций, генерируемых подсистемой анализа данных. При этом, при непосредственной обработке происходит оповещение определенных шаблоном клиентов, о том, что исследуемая система с текущего момента функционирует в нештатном режиме.

Подсистема оповещения позволяет оператору системы оперативно реагировать на сбои в удаленной работе системы и быстро принимать решения об их устранении.

1.3.6 Диспетчеризация

В качестве расширяющей механизм оповещений можно выделить обособленную подсистему диспетчеризации. *Диспетчеризация* – это процесс оперативного контроля, управления, координации какого-либо процесса с использованием оперативной передачи информации между исследуемым объектом и управляющим исследователем.

Система мониторинга реализует подсистему диспетчеризации, если в качестве реакции на возникновение исключительной ситуации она может некоторым образом воздействовать на состояние исследуемой системы. Под воздействием в данном случае понимается удаленное восстановление системы после сбоя и возобновление ее нормальной работы или, если это невозможно, корректное завершение работы исследуемой системы.

1.4 Классификация систем мониторинга

В рамках данной работы предлагается следующая классификация систем мониторинга в сфере информационных технологий: по характеру сетевого взаимодействия и по функционалу (рисунок 1.3).

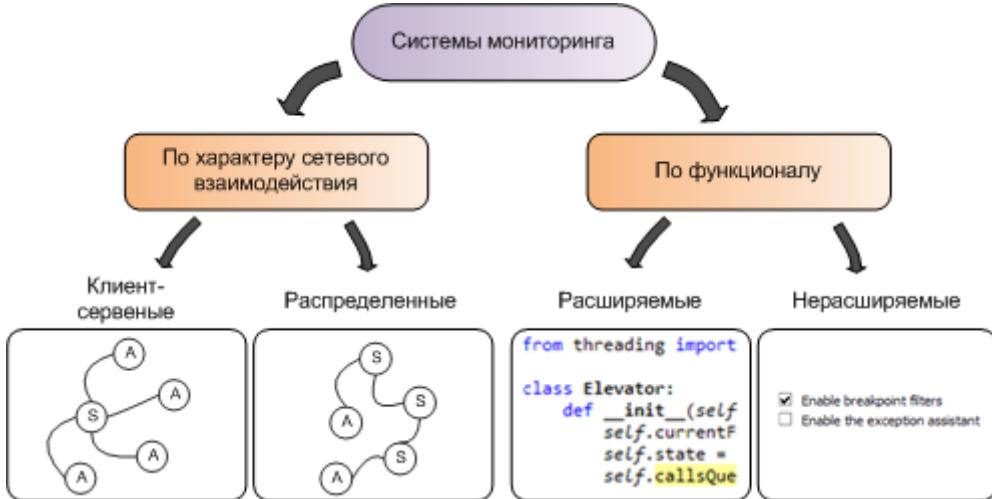


Рисунок 1.3 – Классификация систем мониторинга

По характеру сетевого взаимодействия можно выделить клиент-серверные и распределенные системы мониторинга.

Клиент-серверные или централизованные системы построены по принципу классических сетевых систем с выделенным сервером. В таких системах присутствуют активные и пассивные сущности – агенты и серверы мониторинга соответственно. В качестве примера клиент-серверных систем мониторинга можно привести продукты Zabbix[3], Nagios[5].

В распределенных или децентрализованных системах мониторинга отсутствует понятие сервера, в классическом его понимании. Каждый узел распределенной системы может одновременно являться как сервером, так и агентом мониторинга. Текущее состояние узла и его поведение характеризуются глобальным состоянием распределенной системы [1], которое имеет свойство изменяться под воздействием как внутренних, так и внешних факторов. Единственной в полном смысле распределенной системой мониторинга является проект Ganglia [4].

С точки зрения функционала системы можно выделить системы с расширяемым и нерасширяемым функционалом.

Будем считать, что система мониторинга является системой с расширяемым функционалом, если в её коробочной поставке есть штатные средства и инструменты, позволяющие динамически наращивать функционал

целевой системы. Как правило, подобные инструменты динамического расширения функционала реализованы в виде механизмов разработки и исполнения дополнительных модулей или плагинов системы мониторинга на каком-либо языке программирования. Например, системы Nagios, Zabbix, Mon[6] являются системами с расширяемым функционалом.

Системы мониторинга с нерасширяемым функционалом характеризуются фиксированным, достаточно общим базовым набором функций и возможностей, расширение которого возможно только при участии производителя. Инструментами расширения функционала в этом случае являются пакеты обновления системы, предоставляемые производителем. Вышеупомянутый проект Ganglia, а также системы на базе Zenoss [8] являются примерами решений с нерасширяемым функционалом.

1.5 Требования к системам мониторинга

Применимость и практическая ценность систем мониторинга определяется их способностью адаптироваться к условиям динамически изменяющихся требований, среди которых декларируются требования к функционалу системы, отказоустойчивости и масштабируемости.

Требования к функционалу системы мониторинга опираются на область ее внедрения и последующей эксплуатации. Необходимость изменения функционала возникает, как правило, вследствие динамики внешних требований. Можно условно разделить внешние требования на две группы — технические и правовые. Под техническими требованиями понимаются требования к функционалу системы, основанные нацелях мониторинга, сетевой инфраструктуре, оборудовании или программном обеспечении. Под правовыми требованиями понимаются требования законов государства, лицензионных соглашений, корпоративных уставов.

Под *отказоустойчивостью* понимают способность технической системы сохранять работоспособность и правильно функционировать после отказа некоторых ее компонентов, возможно основных. Применительно к системам

мониторинга можно дать следующее определение понятию отказоустойчивости. Система мониторинга называется отказоустойчивой, если она продолжает функционировать согласно целям мониторинга после отказа любой компоненты или сущности системы. Известно, что повышение отказоустойчивости достигается за счет избыточности или резервирования наиболее значимых ресурсов системы. Наиболее значимой сущностью в архитектуре систем мониторинга является сервер мониторинга. Поэтому, репликация или резервирование серверов мониторинга – это наиболее предпочтительный способ повышения отказоустойчивости системы мониторинга.

Масштабируемость системы мониторинга может измеряться по двум различным показателям. Во-первых, система может быть масштабируемой по отношению к ее размеру, что означает легкость подключения к ней дополнительных клиентов и ресурсов. Во-вторых, система может масштабироваться географически, то есть клиенты и ресурсы могут быть значительно удалены друг от друга в гео-пространстве. Применительно к системам мониторинга, масштабируемость во всех смыслах определяется способностью взаимодействия географически удаленных узлов, а также легкостью подключения новых вычислительных узлов к системе мониторинга.

Рассмотренные выше требования к системам мониторинга позиционируются авторами как основные и дальнейшие рассуждения относительно применимости того или иного класса систем или инструментов будут проводиться с точки зрения этих требований.

1.6 Проблемы эксплуатации систем мониторинга

Основная причина проблем эксплуатации существующих решений в сфере мониторинга заключаются в неспособности последних удовлетворять требованиям к современным системам мониторинга (раздел 1.4). При этом наиболее важным является не столько разовое удовлетворение предъявляемым требованиям, сколько наличие в системе потенциала для возможной ее адаптации к динамике этих требований.

Проблема расширения функционала систем мониторинга была разрешена в системах с поддержкой динамической загрузки модулей, например в системах Zabbix и Nagios, однако остается актуальной для нерасширяемых систем, в частном случае для решений на базе Ganglia. С другой стороны, данная проблема может трактоваться не только как наличие или отсутствие соответствующих механизмов наращивания функционала, но и как уровень их применимости и возможностей. Тогда можно говорить о недостаточной гибкости существующих решений в плане средств расширения функционала, как например, в системах Zabbix и Nagios, где в качестве подобного инструмента используется модель с запуском исполняемых файлов сопровождаемым перехватом стандартных потоков ввода/вывода операционной среды.

Проблема отказоустойчивости характерна только для класса клиент-серверных систем. В распределенных системах она решается на уровне теоретической модели за счет использования методов избыточности, репликации и сериализуемости [1]. Система мониторинга кластеров и гридов Ganglia является хорошим примером действительно распределенной системы.

Проблема масштабируемости системы по отношению к ее размеру также актуальна только для клиент-серверных систем. В распределенных системах мониторинга стоимость подключения новых вычислительных узлов для системы в целом равна нулю благодаря использованию механизмов балансировки нагрузки, а также сокрытию времени ожидания связи. Так при построении системы мониторинга на базе решения Ganglia можно не заботиться о теоретическом пределе количества подключаемых к системе устройств. В противоположность этому, в технической документации клиент-серверные решения Zabbix и Nagios определено максимальное количество обслуживаемых устройств.

1.7 Выводы

Согласно приведенным выше рассуждениям можно сделать вывод о том, что основополагающая проблема эксплуатации современных систем мониторинга

заключается в отсутствии на рынке целого класса комбинированных систем, одновременно объединяющих в себе преимущества как распределенных, так и расширяемых систем мониторинга. Кроме того, современные тенденции развития облачных и кластерных решений в области суперкомпьютерных технологий лишь подтверждают необходимость появления подобных инструментов мониторинга.

Таким образом, рассмотренные выше проблемы эксплуатации систем мониторинга, позволяют сделать вывод о неготовности существующих решений комплексно выполнять выдвинутые к ним требования.

Авторами предлагается проект распределенной системы мониторинга и диспетчеризации процессов гетерогенной среды, которая позволяет обеспечить выполнение перечисленных требований.

2 Модель распределенной системы мониторинга

2.1 Общие положения

Авторами предлагается модель архитектуры распределенной системы мониторинга, которая позволяет обеспечить выполнение рассмотренных в первом разделе требований. Главная идея предлагаемого подхода заключается в использовании механизма разработки и исполнения дополнительных модулей в процессе решения задач мониторинга, а также свойств распределенных систем в процессе эксплуатации.

Предлагаемая методология построения распределенных систем мониторинга является обобщением и объединением существующих классов систем в сфере мониторинга – распределенных систем с расширяемым функционалом.

2.1 Базовая теоритическая модель

Базовая теоретическая модель описывается с помощью следующих понятий (рисунок 2.3):

- а) вычислительный узел;
- б) служба мониторинга;
- в) хранилище данных;
- г) задача мониторинга.

Под *вычислительным узлом* далее будем понимать программно-аппаратное устройство, в память которого может быть загружен и затем исполнен код какой-либо сущности мониторинга.

Служба мониторинга, запущенная на определенном узле, представляется активной сущностью, непрерывно наблюдающей за его состоянием и сохраняющей сообщения об изменении этого состояния в хранилище данных.

Хранилище данных представляется пассивной сущностью, предоставляющей службам ресурсы для приема сообщений, их последующей обработки и хранения.

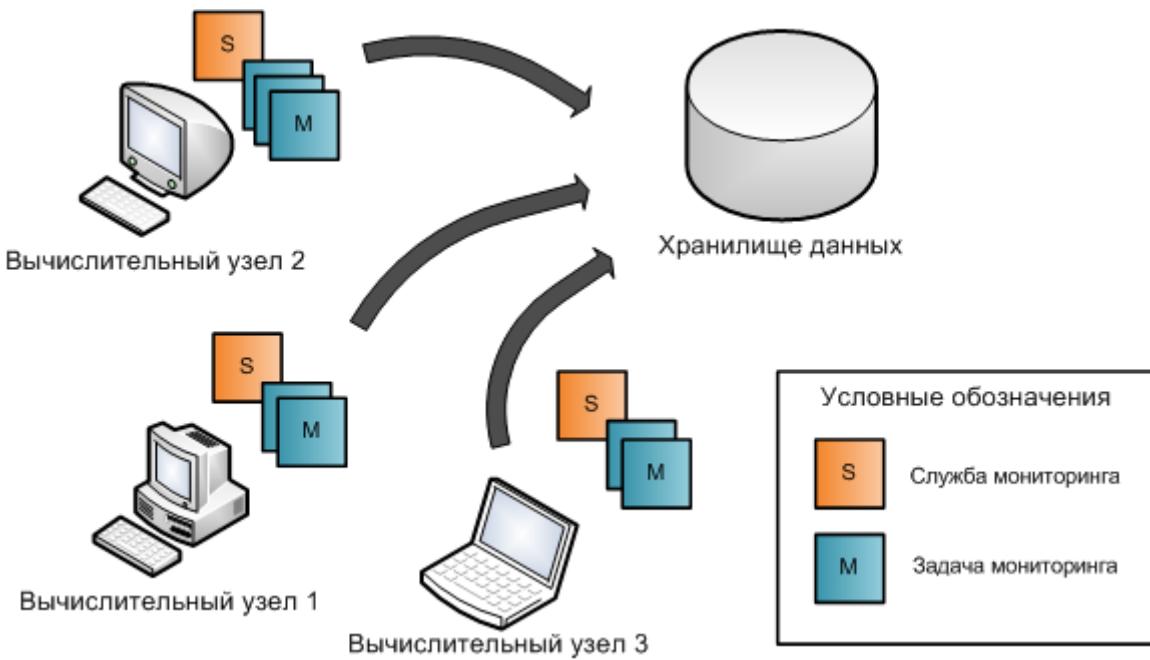


Рисунок 2.3 – Базовая модель

Фактически служба мониторинга и хранилище данных являются основными компонентами любой современной системы мониторинга за исключением небольшого количества программных продуктов, не реализующих клиент-серверную или иную известную сетевую архитектуру, но являющихся инструментами мониторинга (например, утилита ping).

Задача мониторинга представляет собой шаблонную проблему получения и анализа некоторой информации о состоянии удаленного узла.

Можно ввести отношение между целью (раздел 1.1 «Понятие систем мониторинга») и задачей мониторинга. Какая-либо цель мониторинга включает в себя одну или более задач мониторинга. При этом какая-либо задача мониторинга может одновременно реализовывать несколько целей мониторинга.

2.2 Модуль мониторинга

Под *модулем мониторинга* далее будем понимать абстракцию (рисунок 2.4), характеризующуюся:

- возможностью исполнения в операционной среде;
- входными данными, передаваемыми исполняющей системой;

- в) выходными данными, возвращаемыми исполняющей системе;
- г) интерфейсом, задающим правила исполнения модуля;
- д) реализацией, представляющей собой программный код, воплощающий функционал модуля.

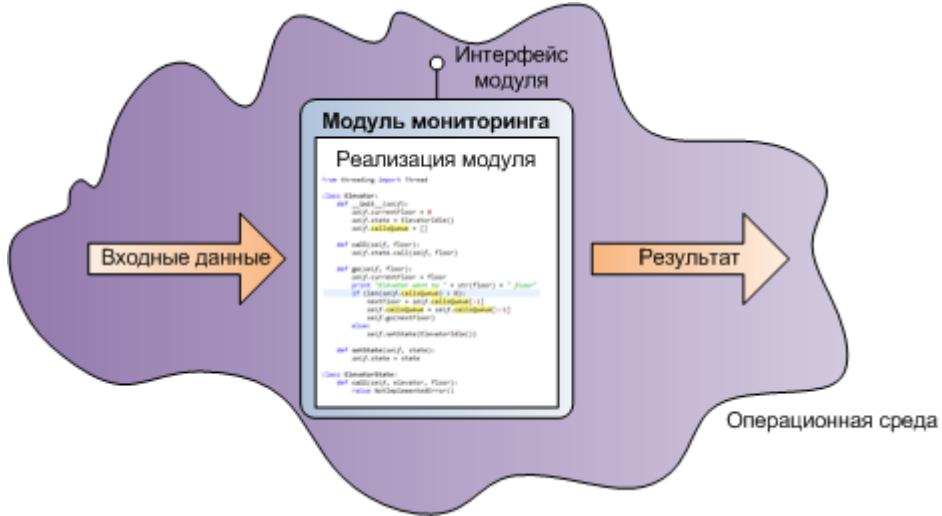


Рисунок 2.4 – Абстракция модуля

Понятие модуля мониторинга является следствием отображения задачи мониторинга из предметной области в программную среду.

2.3 Система исполнения

Система исполнения модулей мониторинга (рисунок 2.5) реализует генерацию кода каркаса и исполнение модулей мониторинга с использованием ресурсов операционной среды, а также является промежуточным слоем между модулем мониторинга и службой, в рамках которой он запускается. Данный слой позволяет разрабатывать модули без учета специфики физического расположения служб мониторинга, игнорируя такие особенности, как адресация и топология сети.

Система исполнения является определяющей компонентой в модели службы мониторинга и от ее реализации зависит эффективность и применимость системы в целом. Кроме того, система исполнения является платформенно-зависимой частью системы и должна разрабатываться под определенный набор операционных и аппаратных платформ.

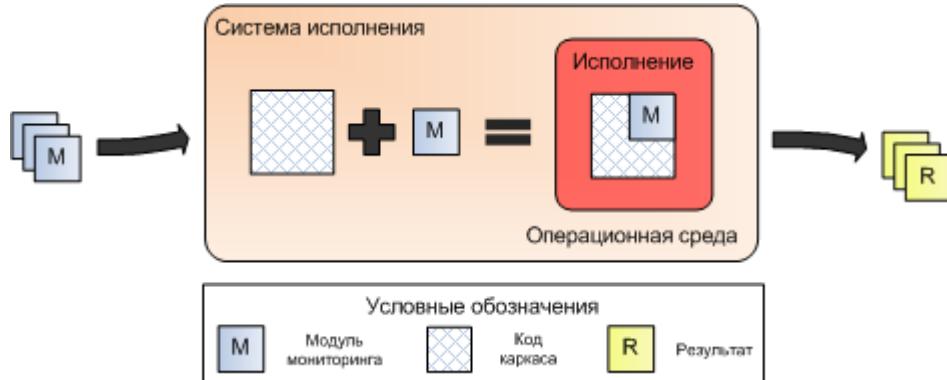


Рисунок 2.5 - Система исполнения

2.4 Код каркаса

Код каркаса (рисунок 2.6) генерируется системой исполнения на основании текущего глобального состояния распределенной системы и содержит следующие конструкции:

- а) инициализации окружения;
- б) создания экземпляра модуля мониторинга;
- в) исполнения экземпляра;
- г) передача параметров и ожиданием возвращаемого результата.

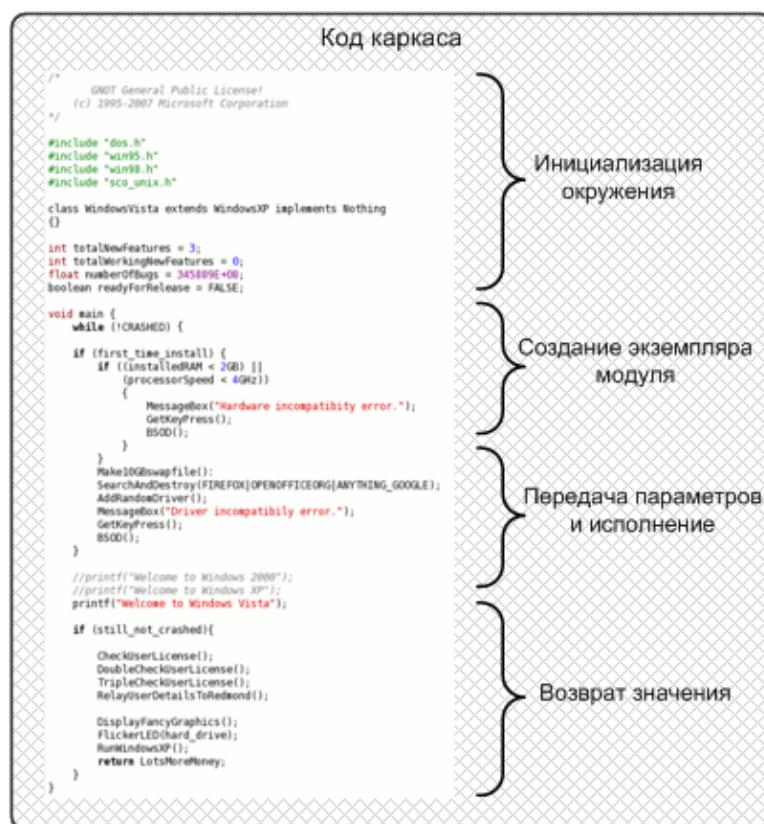


Рисунок 2.6 - Код каркаса

2.5 Прикладной интерфейс программирования

Модули мониторинга разрабатываются в терминах предметной области с использованием прикладного интерфейса программирования (API) — высокоуровневого объектно-ориентированного набора инструментов. Прикладной интерфейс программирования (рисунок 2.7) является промежуточным слоем между модулем мониторинга и операционной средой, в которой он запущен. API призван сосредоточить программиста на решаемой задаче мониторинга, скрыв от него подробности реализации сложных моментов, таких как распределенная коммуникация с сервером, маршализация/демаршализация параметров и возвращаемого результата модуля, системные вызовы операционной системы.

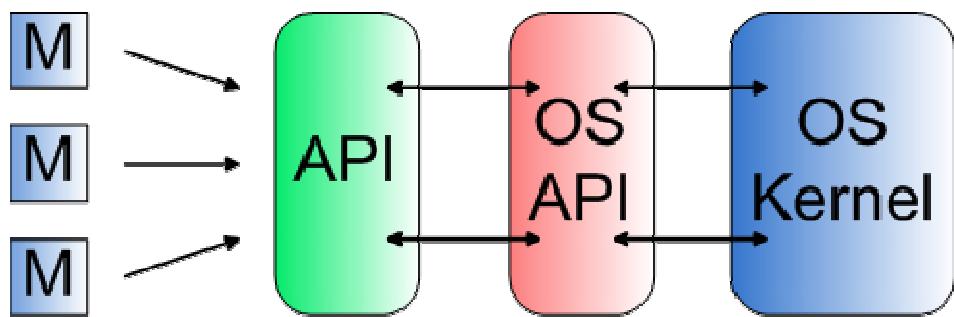


Рисунок 2.7 – Взаимодействие модулей и ОС через API

2.6 Состояние распределенной системы мониторинга

Известно понятие глобального состояния [1], в соответствии с которым распределенная система функционирует в данное время (рисунок 2.8). В классической трактовке состояние определяется графом связности узлов, расположением запущенных экземпляров модулей и нагрузкой на узлы. В предлагаемой модели сущность распределенного модуля представляет служба мониторинга. Это придает ей некоторые особенности элемента распределенной системы, например:

- а) масштабируемость — возможность запуска дополнительного экземпляра;

- б) сериализуемость — возможность сохранения текущего состояния службы;
- в) переносимость — возможность переноса службы в распределенной среде с сохранением ее внутреннего состояния.

Служба мониторинга характеризуется своим внутренним непротиворечивым состоянием – активным или пассивным. Активное состояние наделяет службу дополнительными обязанностями по отношению к соседним узлам: планирование запусков модулей мониторинга; мониторинг и диспетчеризация процессов исполнения модулей мониторинга; предоставление промежуточного хранилища для пересылаемых сообщений.

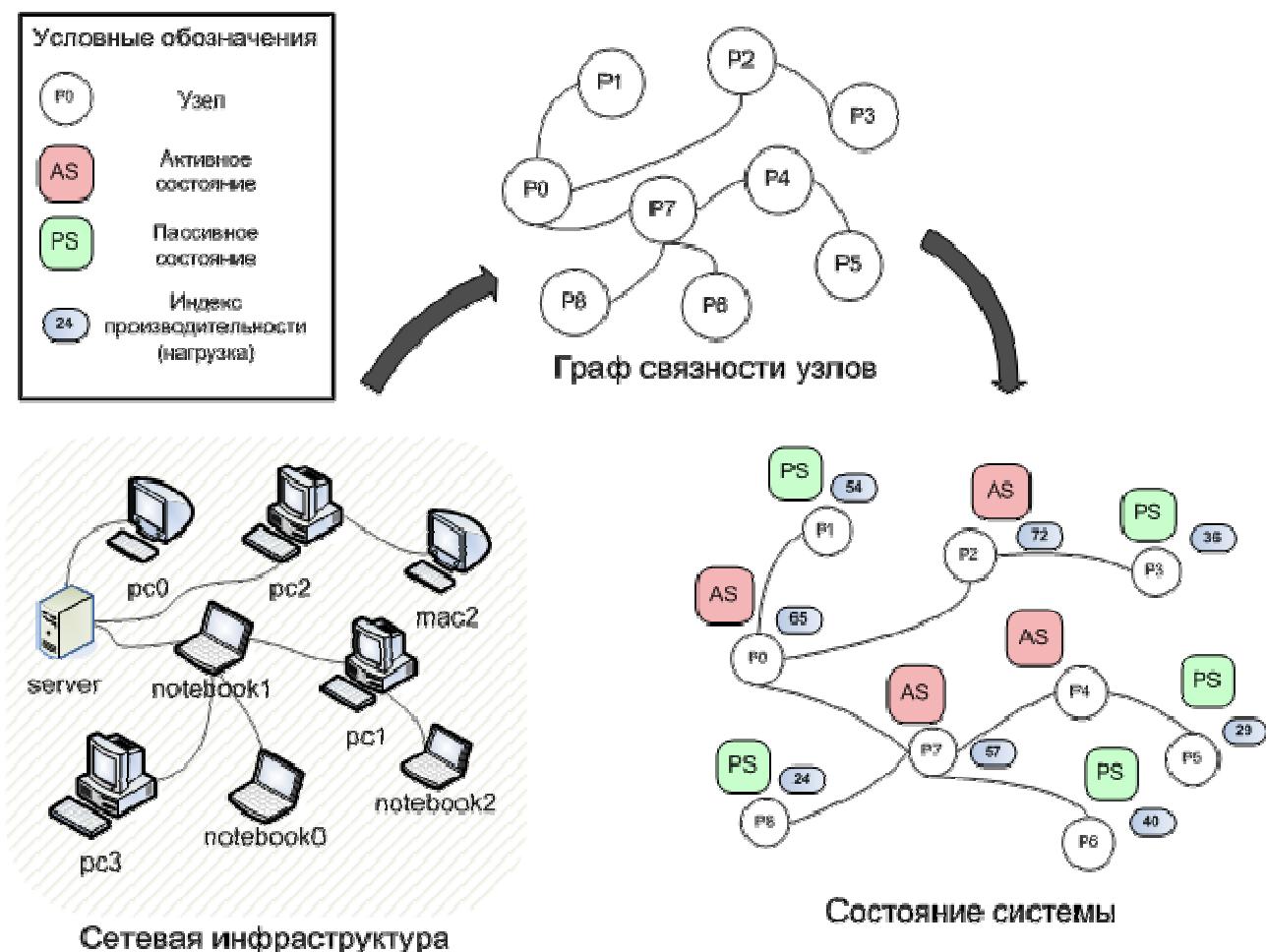


Рисунок 2.8 - Состояние распределенной системы

В предлагаемой модели роль нагрузку на узел играет индекс производительности — целое положительное число, определяющее количество

свободных вычислительных ресурсов узла по некоторой абсолютной или относительной шкале. Индекс производительности узла совместно с установленным пороговым значением являются рычагами воздействия на глобальное состояние распределенной системы мониторинга.

Службы, запущенные на узлах с индексом производительности ниже порогового значения, подвергаются масштабированию (запуску дополнительных экземпляров, сопровождаемому балансировкой нагрузки), в результате чего распределенная система переходит в более производительное состояние (рисунок 2.9).

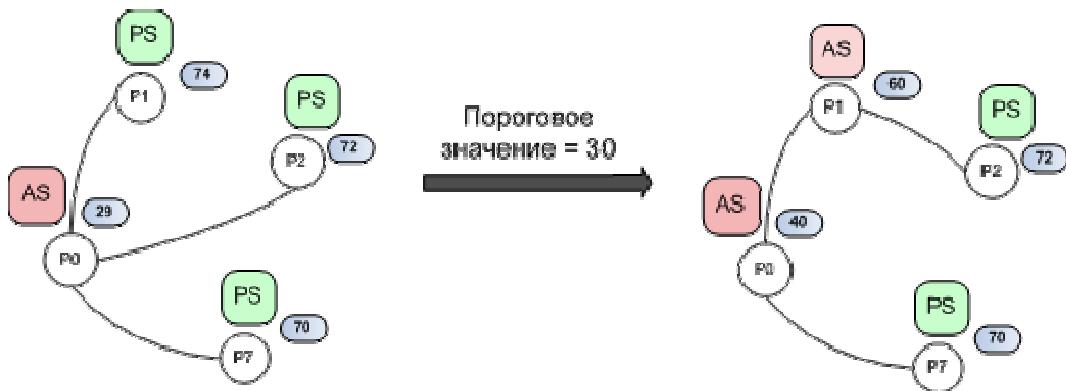


Рисунок 2.9 – Изменение состояния системы

3 Реализация системы

3.1 Служба мониторинга

3.1.1 Структура службы мониторинга

Служба мониторинга (рисунок 3.1) представляет собой программный комплекс, обеспечивающий использование ресурсов вычислительной среды, адресацию, поддержание поведения распределенной системы мониторинга (модулей мониторинга, распределенной коммуникации, программной системы в целом).

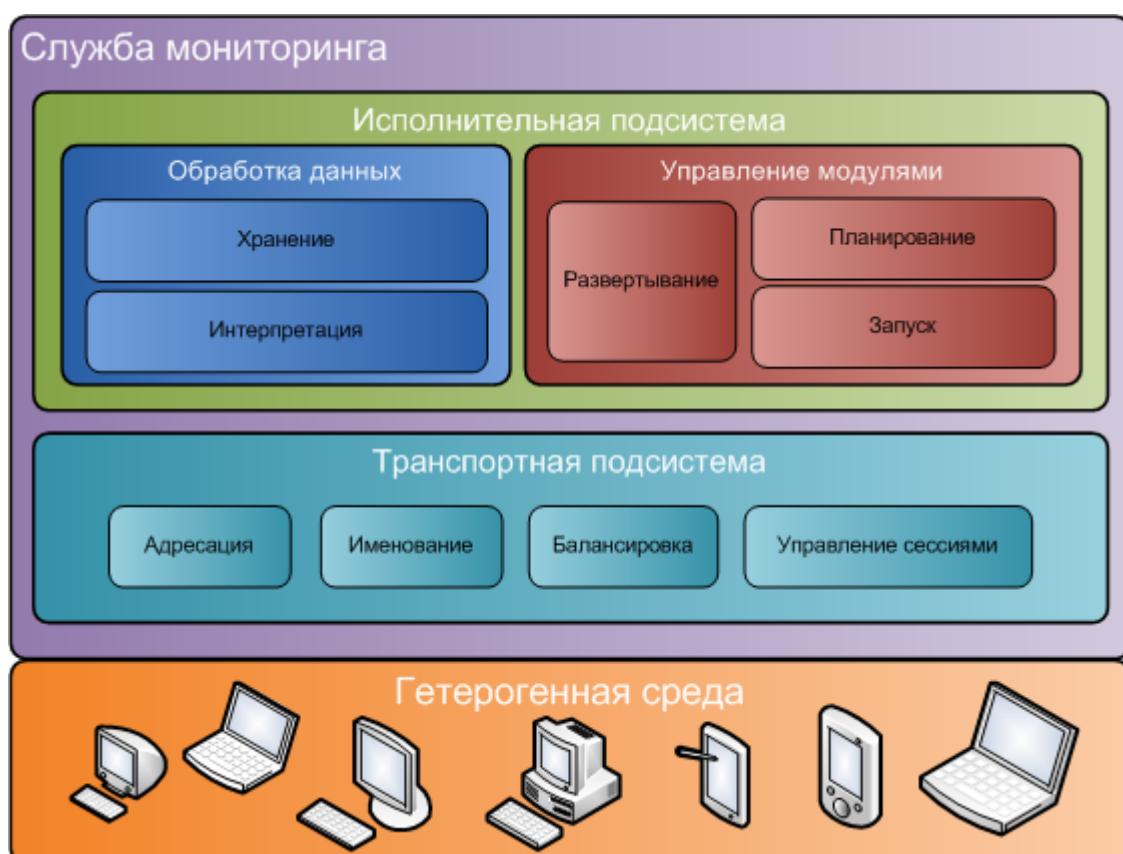


Рисунок 3.1 - Служба мониторинга

Служба мониторинга распределенной системы состоит из двух основных подсистем:

- подсистемы исполнения, позволяющей планировать и запускать модули мониторинга, а также получать результаты исполнения модулей и сохранять их в хранилище данных;
- транспортной подсистемы, реализующей сетевую модель распределенной системы и включающей в себя механизмы именования объектов, удаленной коммуникации, адресации и балансировки нагрузки.

Служба мониторинга представляет собой распределенное приложение, а, следовательно, должна устойчиво функционировать в гетерогенной вычислительной среде.

3.1.2 Выбор средств реализации

3.1.2.1 Модель программирования

В процессе выбора средств реализации службы мониторинга были рассмотрены популярные на сегодняшний день технологии построения распределенных систем. Кроме того, были учтены особенности реализации существующих решений в области мониторинга. Авторами были рассмотрены следующие технологии разработки распределенных систем:

- модель программирования на распределенной памяти или механизмы передачи сообщений (PVM, MPI);
- распределенные системы объектов (CORBA);
- нетрадиционные языки программирования для распределенных систем (Erlang);
- библиотеки промежуточного слоя (Ice ZeroC).

В итоге, была выбрана объектно-ориентированная платформа среднего слоя Ice (The Internet Communication Engine) от компании ZeroC в силу доминирующего превосходства над аналогами. Платформа Ice снабжена инструментами, API и библиотеками для разработки объектно-ориентированных клиент–серверных приложений. Ice-приложения могут быть написаны на различных языках программирования (Java, C++, Python, C#, Ruby), запущены под

различными операционными системами (Windows NT, Linux, MacOS OS) и аппаратными платформами, а также могут взаимодействовать, используя разнообразные сетевые технологии. В общем случае Ice позиционируется как инструмент RPC (Remote Procedure Call), который достаточно прозрачно применять на практике. Большое количество компаний по всему миру, таких как Skype, HP, Silicon Graphics используют технологию Ice в своих проектах.

Платформа Ice обладает следующими особенностями и преимуществами:

- объектно-ориентированная семантика;
- поддержка синхронных и асинхронных вызовов;
- аппаратная независимость;
- языковая независимость;
- операционная независимость;
- безопасность;
- доступность исходного кода;

Кроме того, используемый в Ice объектно-ориентированный подход позволяет переиспользовать уже известные шаблоны проектирования параллельных многопоточных систем без каких-либо исключений.

3.1.2.2 Терминология модели программирования

Для более детального понимания реализации рассмотрим основные понятия и сущности, представленные платформой Ice. Основные термины модели программирования трактуются с помощью типовых понятий из теории по клиент-серверным системам.

Можно выделить два вида программных агентов в платформе Ice:

- а) клиент — активная сущность, запрашивающая некоторые ресурсы у сервера;
- б) сервер — пассивная сущность, предоставляющая некоторые ресурсы клиенту.

На практике редко встречаются «чистый» сервер или «чистый» клиент. Чаще всего это смешанный клиент-сервер, который одновременно и запрашивает и предоставляет ресурсы.

Любая сущность, запущенная в распределенной системе Ice, определяется так называемым Ice-объектом. Ice-объект – это абстракция, характеризующаяся следующим:

- локальными или удаленным адресным пространством;
- возможностью реакции на удаленный запрос;
- поддержкой репликации;
- несколькими интерфейсами;
- публичными методами интерфейсов имеющих как входные, так и выходные параметры;
- уникальным идентификатором, не совпадающим с любым другим идентификатором объекта в распределенной гетерогенной среде.

Для непосредственной распределенной коммуникации объектов платформа Ice реализует специальные механизмы транспортного уровня, инкапсулированные в объектах типа прокси. Практически, Ice-прокси соответствует одному Ice-объекту и предоставляет для него механизмы и примитивы для организации удаленных вызовов процедур. Прокси – это артефакт, который локален для клиентского адресного пространства. Ice-прокси инкапсулирует следующую информацию, о действиях, совершаемых платформой привызове удаленной процедуры клиентом:

- а) определяет местоположение Ice-объекта;
- б) активирует сервер, если он не запущен;
- в) активирует Ice-объект на сервере;
- г) передает входные параметры Ice-объекту;
- д) ждет, когда операция закончится;
- е) передает возвращаемые значения или генерирует исключение.

Кроме того, Ice-прокси содержит:

- а) адресную информацию, которая позволяет клиентской стороне соединиться с нужным сервером;
- б) идентификатор объекта, который является целью запроса;
- в) дополнительный идентификатор, который определяет интерфейс объекта, к которому прокси обращается.

Стоит также рассмотреть виды запросов на диспетчеризацию и виды диспетчеризации вызовов.

Платформа среднего слоя Ice поддерживает следующие виды запросов на диспетчеризацию:

- синхронный вызов, при котором клиент, совершивший вызов, повисает на время выполнения процедуры, пока она не закончится;
- асинхронный вызов, сопровождаемый передачей объекта обратного вызова;
- односторонний вызов метода с организацией одностороннего потока сетевого трафика;

Аналогично запросам на диспетчеризацию можно выделить следующие виды диспетчеризации методов:

- синхронная диспетчеризация, при которой серверный поток повисает в ожидании завершения процедуры;
- асинхронная диспетчеризация, эквивалентная асинхронному вызову методов;

Для описания удаленных интерфейсов объектов Ice использует специализированный язык описания спецификаций – Slice (SpecificationLanguageforIce).

Каждый Ice-объект имеет интерфейс с конечным набором операции. Интерфейсы, операции и типы данных, которыми обмениваются сервер и клиент, должны быть описаны на языке Slice. Slice позволяет описывать поведение сервера и клиента, не опираясь на какой-либо язык программирования, благодаря этому написанный код на Slice компилируется в код любого из поддерживаемых платформой языков программирования.

3.1.2.3 Язык программирования

Платформа среднего слоя Ice поддерживает почти все современные языки программирования, среди которых можно отметить наиболее популярные:

- нативные (C++, Objective C);
- управляемые (Java, C#, VB.NET);
- динамические/интерпретируемые (Python, PHP).

Для выбора инструмента программирования рассмотрим основные выдвигаемые к нему требования:

- достаточно высокая производительность языка или платформы;
- кроссплатформенность;
- поддержка ООП-семантики;
- наличие современных и эффективных средств разработки;
- доступность языка или платформы;
- богатая библиотека стандартных модулей и классов.

Кроме того, можно выделить еще один важный критерий выбора – скорость разработки и простота внесения изменений в программный код.

Динамически интерпретируемые языки, такие как Python и PHP, не удовлетворяют требованиям к производительности. Безусловно, современные технологии построения интерпретаторов позволяют им добиваться сравнимой с нативным кодом производительности, однако ядро службы мониторинга является бутылочным горлышком системы и может существенно влиять на поведение и скорость работы всей системы в целом. Поэтому нами рассматривались два варианта – языки Java и C++. Языки на платформе .NET не рассматривались из-за отсутствия кроссплатформенной реализации.

С одной стороны оба языка предоставляют программисту сравнительно одинаковый набор возможностей (ООП, стандартная библиотека классов и модулей). С другой – программы, написанные на Java и на C++, показывают абсолютно разную, практически несравнимую производительность.

В конечном счете, нами была выбрана платформа Java. Решающим фактором, определившим наш выбор, стала скорость разработки, которая, как известно, на Java значительно выше, чем на C++. Более того, современные виртуальные машины платформы Java (OracleHotspot, OpenJDK) со встроенными компиляторами динамического кода (JIT) показывают отличную производительность, порой превосходящую нативное исполнение в разы а на порядки. Такой прирост производительности объясняется в первую очередь механизмами динамического профилирования и сборки мусора, которые идеологически недоступны в нативных компиляторах.

Помимо вышеперечисленного, для Java существует большое количество удобных и эффективных сред разработки (Eclipse, Netbeans), отладки, тестирования, а также свободных переносимых библиотек для решения широкого круга прикладных задач.

Безусловно, нельзя отвергать тот факт, что участники проекта при выборе языковой платформы основывались на собственный опыт разработки программных систем, где чаще всего применялась платформа Java.

3.1.3 Ядро системы

3.1.3.1 Общее описание

Ядро службы мониторинга (рисунок 3.2) реализует базовую, динамически расширяемую программную платформу, в рамках которой запускаются и функционируют основные подсистемы службы. Кроме того, ядро обеспечивает работу загружаемых компонентов службы, а также содержит базовые механизмы и примитивы для их взаимодействия и синхронизации.

Как уже отмечалось, ядро представляет собой динамически расширяемую программную модель, функционал которой может изменяться в процессе эксплуатации, посредством загрузки или выгрузки дополнительных компонентов ядра – так называемых *драйверов ядра*.

Функционирование ядра системы реализовано в терминах модели «Цикл событий», смысл которой заключается в бесконечной обработке событий,

приходящих системе от внешних клиентов. В качестве внешних клиентов системы выступают драйверы ядра, каждый из которых реализует определенную часть общего поведения и функционала конечной системы.

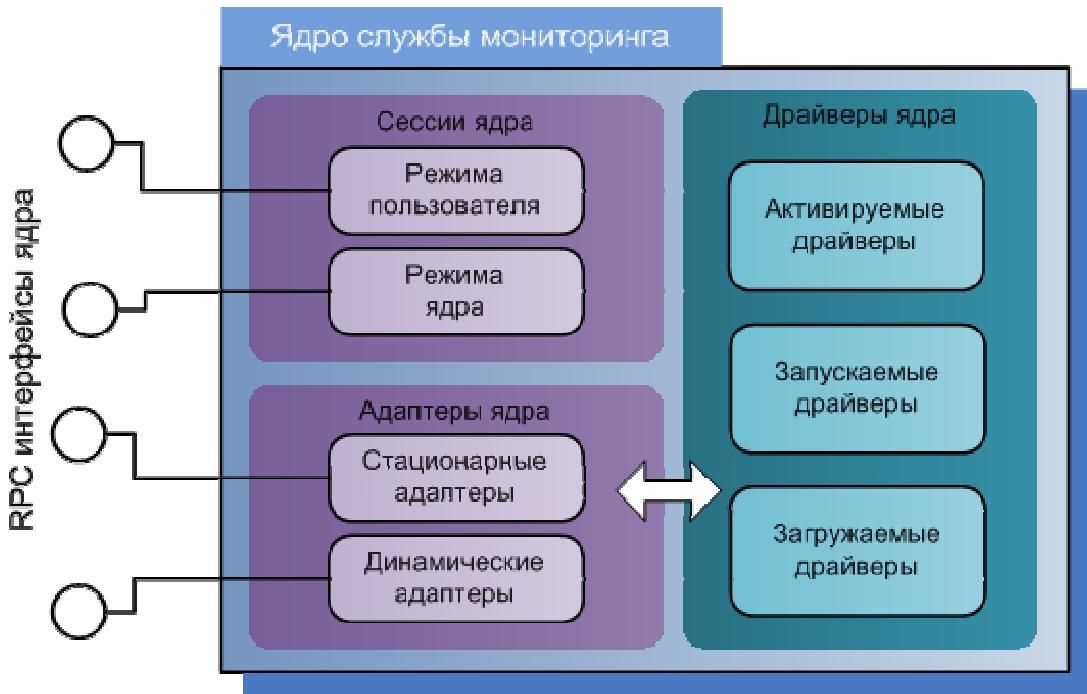


Рисунок 3.2 – Ядро службы мониторинга

Взаимодействие драйверов не осуществляется напрямую. Вместо этого используется генерация, обработка и передача специальных событий ядру. Событие ядра инкапсулирует тип случившейся внутрисистемной ситуации и содержит необходимые параметры и структуры для ее корректной обработки.

Для обработки событий используются обработчики ядра. Ядро имеет несколько обработчиков, каждый из которых соответствует определенному состоянию ядра.

Помимо модели «Цикл событий», ядро реализует парадигму «Конечный автомат» (Finite State Machine). Проще говоря, ядро характеризуется своим внутренним состоянием и может переходить из состояния в состояние при обработке некоторого внутрисистемного события.

Основная идея предлагаемого подхода для разработки ядра распределенной службы мониторинга заключается в так называемых публичных драйверах. Помимо основных драйверов системы, именуемых в дальнейшем

приватными, существуют дополнительные драйверы – публичные. Приватные драйвера могут использоваться только тем ядром, в адресное пространство которого они загружены, в то время как публичные могут использоваться любыми другими удаленными ядрами, запущенными в гетерогенной среде.

Для придания драйверу ядра публичности достаточно реализовать для него, так называемый *адаптер* драйвера ядра. Адаптер драйвера ядра предоставляет внешним клиентам RPC интерфейс для синхронных и асинхронных вызовов публичных методов драйвера. Такая модель применяется для взаимодействия служб, запущенных в различных адресных пространствах.

Кроме того, для удаленного взаимодействия используются *сессии*. Сессия представляет собой набор доступных адаптеров ядра с публичными интерфейсами. Существуют сессии режима ядра и сессии режима пользователя. Сессии режима ядра устанавливаются между удаленными ядрами. Сессии режима пользователя устанавливаются между ядром и панелью управления.

3.1.3.2 Архитектура ядра

Ядро представляет собой автономный поток исполнения, реализующий модель «Цикла событий». Ядро содержит базовые механизмы и примитивы, необходимые для работы системы, такие как обработка событий, изменение состояния системы, управление драйверами/адаптерами и управление сетевой подсистемой.

Ядро также содержит основные таблицы и кэши системы:

- а) пул событий ядра;
- б) таблица подключенных дочерних узлов;
- в) таблица подключенных родительских узлов;
- г) кэш исследованных узлов;
- д) таблица драйверов ядра;
- е) таблица адаптеров ядра;
- ж) таблица фильтров ядра;
- з) таблица наблюдателей ядра.

В каждый момент времени ядро находится в определенном состоянии, менять которое способен только его текущий обработчик, который однозначно определяется состоянием. Кроме того, изменение любых внутренних структур ядра разрешено только потоку ядра. В противном случае – при попытке несанкционированного изменения состояния любым внешним потоком, генерируется внутреннее исключение ядра.

Ядро управляет двумя основными сетевыми адаптерами системы – первичным и вторичным. Сетевые адаптеры ядра используются для реализации транспортного уровня системы – механизма удаленного вызова процедур (RPC).

Ядро службы мониторинга характеризуется внутренним непротиворечивым контекстом ядра, который содержит основную информацию о его текущем состоянии:

- а) идентификатор ядра – 32-х байтовая последовательность, однозначно идентифицирующая ядро в гетерогенной среде (раздел 3.1.3.4 «Уникальный идентификатор узла»);
- б) прокси первичного и вторичного адаптера, необходимые для установления соединения между ядрами, запущенными в различных адресных пространствах;
- в) индекс производительности узла – целое положительное число, определяющее текущую производительность узла по некоторой шкале;
- г) состояние ядра – текущее состояние ядра;
- д) список дочерних подключенных узлов, используемый системой для адекватной оценки производительности всей распределенной системы в целом;
- е) список родительских узлов, используемый исполнительной подсистемой для управления расписанием и запуском модулей мониторинга;
- ж) базовая информация о вычислительном узле (тип операционной системы, имя и IP-адрес компьютера в сети).

3.1.3.3 Исключения ядра

Для соблюдения парадигмы защищенного программирования авторами была разработана модель исключений ядра.

В предлагаемой архитектуре ядра распределенной службы мониторинга существует два вида внутрисистемных исключений:

- а) нативные исключения ядра;
- б) внешние исключения, генерируемые драйверами.

Нативные исключения ядра генерируются самим ядром при следующих ситуациях:

- а) ошибка инициализации и deinициализации ядра;
- б) некорректная обработка события ядра;
- в) ошибка загрузки и выгрузки драйвера ядра;
- г) недетерминированный переход между состояниями ядра;
- д) обработка внешних системных исключений виртуальной машины или операционной системы.

В отличие от нативных исключений, внешние исключения обрабатываются ядром как обычные события (раздел «3.1.3.10 События ядра»). Источниками внешних исключений могут служить следующие ситуации:

- а) некорректная работа локального или удаленного драйвера ядра;
- б) ошибка в транспортной подсистеме;
- в) ошибка в исполнительной подсистеме службы.

Внедрение в программную модель механизма обработки и генерации исключений и применение принципов защищенного программирования позволило обеспечить систему необходимым уровнем отказоустойчивости и предсказуемости.

3.1.3.4 Пул ядра

Поток ядра реализует механизм бесконечной обработки событий, который генерируют ядру драйвера. Событие при генерации попадает в пул ядра - потокобезопасную очередь с фильтрацией и без планирования. Это значит, что

события обрабатываются ядром по принципу FIFO. Благодаря использованию потокобезопасного пула, поток ядра имеет особенность «засыпать» при условии, что пул пустой и в текущий момент нет запроса на обработку. При этом первое поступившее событие в очередь его «разбудит» и основной цикл будет продолжен.

Это позволяет экономить ресурсы системы в сетях с небольшой нагрузкой.

Перед непосредственной обработкой события происходит этап фильтрации событий ядра (раздел 3.1.3.4 «Фильтры пула ядра»). С практической точки зрения фильтрация представляет собой последовательное применение цепочки фильтров на каждое событие. В результате – событие может быть отфильтровано либо пропущено.

Псевдокод основного цикла потока ядра для обработки пула событий изображен на рисунке 3.3.

```
for (;started; ) {
    for (; !pool.isEmpty() && started; ) {
        Event event = pool.poll();

        eventFiltered = false;

        for (Filter filter : filters) {
            if (FilterAction.REJECT == filter.accept(event)) {
                eventFiltered = true;
                break;
            }
        }
        if (!eventFiltered) {
            handler.handle(event);
        }
    }
    waitForNotify();
}
```

Рисунок 3.3 – Псевдокод потока ядра

3.1.3.5 Фильтры пула ядра

При проектировании модели поведения ядра стало ясно, что прямая реализация классической модели обработки событий имеет свои определенные

недостатки и требует формальных доработок и изменений для данной архитектуры. В первую очередь, это касается пула ядра.

В процессе эксплуатации экспериментального прототипа, опытным путем, было выяснено, что не все поступившие в пул ядра события должны быть обработанными.

При определенной последовательности событий в пуле модель переходов между состояниями ядра становилась недетерминированной. Особенно это проявлялось в многопоточной среде исполнения. Для придания ядру детерминированного поведения, нами был разработан механизм фильтрации событий из пула ядра.

Фильтры пула ядра применяются для исключения определенного класса событий из пула до момента их обработки. Фильтры ядра образуют последовательную цепочку, по которой проходит каждое событие из пула. Событие считается отфильтрованным, если хотя бы один фильтр из цепочки сработал.

В текущей реализации доступно два фильтра ядра:

- а) фильтр переходов (ToogleFilter);
- б) фильтр UDP сообщений (UDPFILTER).

Фильтр переходов ориентирован на фильтрацию сообщений об изменении состояния ядра. Семантика работы фильтра подразумевает удаление из пула всех неразрывных последовательностей событий об изменении состояния ядра кроме последнего.

Фильтр UDP сообщений исключает системные сообщения от удаленных служб, если ядро находится в активном состоянии (раздел 3.1.3.5 «Состояния и обработчики ядра»).

Архитектурно, фильтры пула ядра представляют собой реализацию шаблона «Посетитель/Visitor» [link] (диаграмма классов на рисунке 3.4).

Реализация механизмов фильтрации пула событий позволило внести определенную ясность и предсказуемость в поведение ядра. В частности,

наибольший эффект от внедрения механизмов фильтрации ожидался в исключении недетерминированности переходов между состояниями ядра.

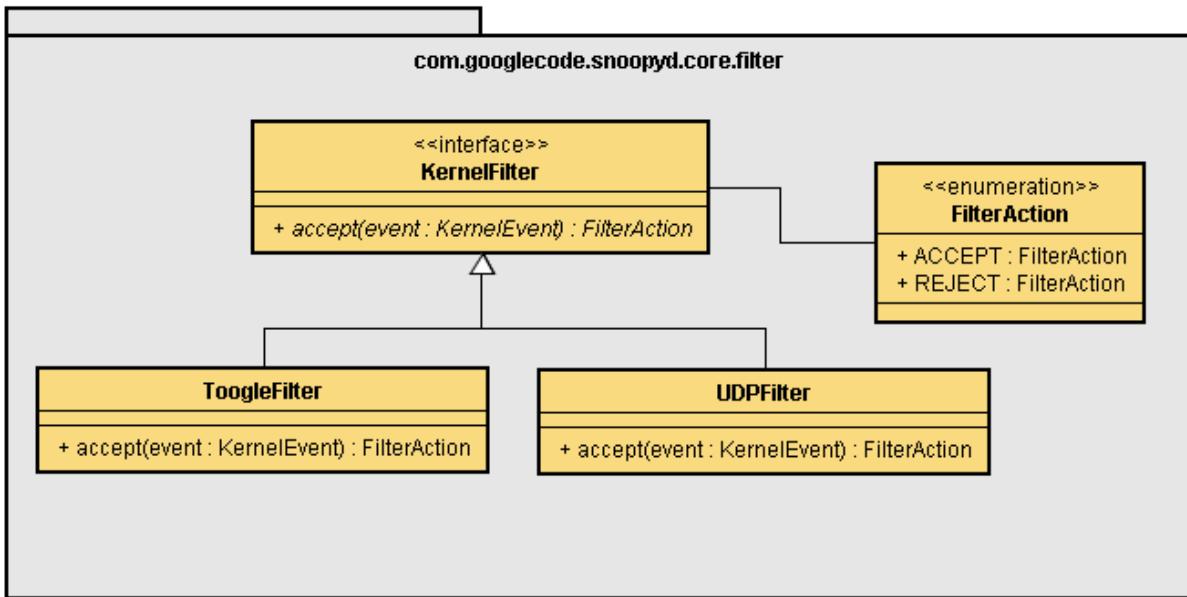


Рисунок 3.4 – Диаграмма классов пакета фильтров ядра

3.1.3.6 Состояния и обработчики ядра

Для реализации поведения ядра службы в терминах модели конечного автомата авторами были спроектированы и реализованы конечные множества состояний и событий ядра. Существует пять различных состояний ядра системы:

- активное (active state);
- пассивное (passive state);
- сетевое (online state);
- автономное (offline state);
- неопределенное (suspense state).

Как уже отмечалось, состояние ядра однозначно определяет обработчик, которым будут обрабатываться события, приходящие ядру из внешней среды. Таким образом, можно утверждать, что в каждый момент времени состояние ядра определяет его поведение.

С точки зрения реализации, и состояние и обработчик ядра представляются шаблоном проектирования «Состояние/State» [link] (рисунок 3.5).

При этом оба интерфейса содержат лишь по одному публичному методу. В случае с состоянием – это метод получения текущего обработчика, в случае с обработчиком – это метод обработки следующего события ядра.

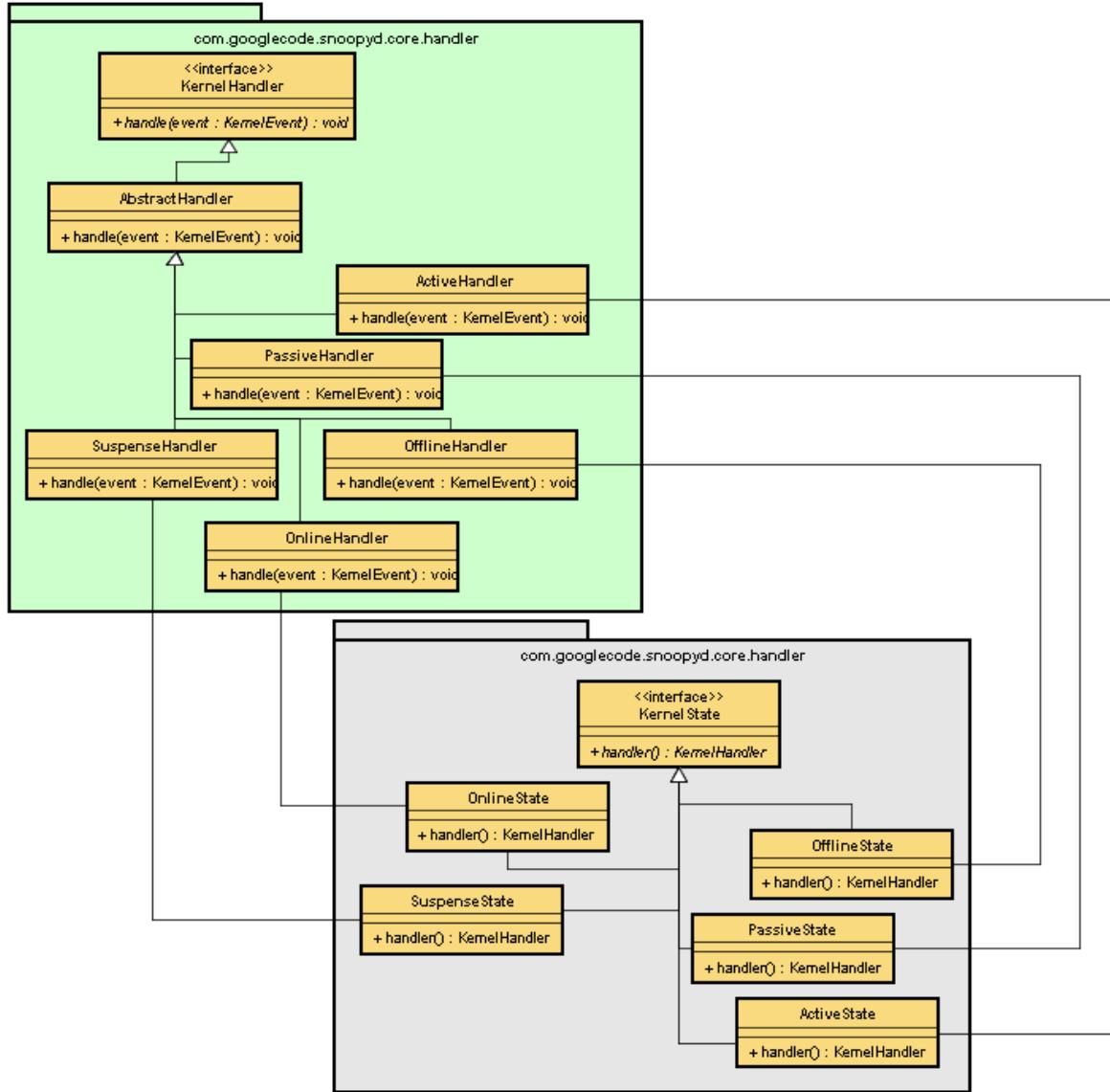


Рисунок 3.4 – Диаграмма классов обработчиков и состояний

Переход из состояния в состояние осуществляется только при обработке определенного класса событий. На рисунке 3.6 рассмотрены основные циклы смены состояний у ядра. Более детально про переходы между состояниями рассмотрены в разделах 3.1.3.1 «События ядра» и 3.1.3.12 «Поведение ядра».

При изменении своего состояния ядро оповещает всех заинтересованных в этом событии драйверов. Такие драйвера реализуют специальный интерфейс, о котором подробно будет сказано в разделе 3.1.3.6 «Наблюдатели ядра».

Реализация модели поведения ядра в терминах конечных автоматов оказалась хорошей практикой. Отладка, тестирование и разработка пределенной системы становится более очевидной и понятной для программиста при применении данного подхода.

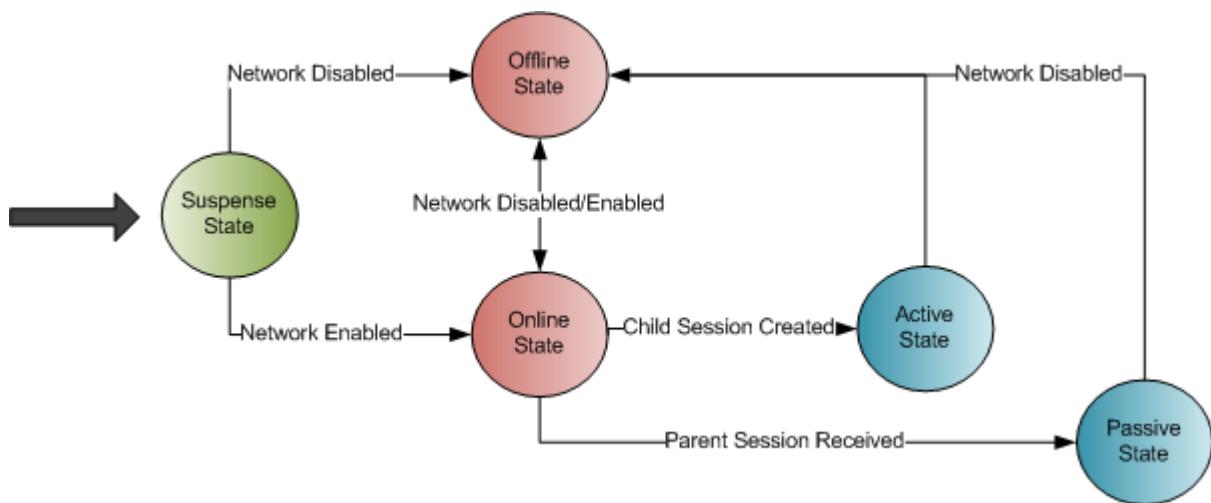


Рисунок 3.5 – Состояния ядра

3.1.3.7 Наблюдатели ядра

В процессе реализации драйверов ядра (раздел 3.1.3.7 «Драйверы ядра»), стало понятным, что любое изменение состояния ядра должно явно или не явно отражаться на дальнейшем поведении драйверов. Иными словами, каждый в отдельности взятый драйвер ядра должен функционировать только тогда, когда ядро находится в конкретном, пригодном для нормальной работы этого драйвера состоянии.

Для реализации подобного поведения нами был спроектирован и разработан интерфейс наблюдателя ядра, который должны реализовывать все драйвера, нуждающиеся в зависимом от состояния поведении. Интерфейс наблюдателя ядра содержит один публичный метод, оповещающий драйверы об

изменении состояния ядра и передающий драйверам признак этого нового состояния.

Можно выделить следующие особенности, которыми характеризуется драйвер, реализующий интерфейс наблюдателя:

- а) отложенный (lazy) запуск/останов или активация/деактивация;
- б) зависящее от состояния ядра специфичное поведение.

С точки зрения реализации наблюдатель ядра представляет собой шаблон проектирования «Наблюдатель/Observer»[link]. Тогда драйверы, реализующие интерфейс наблюдателя, в терминах шаблон являются подписчиками.

Более подробно о конкретных драйверах, реализующих интерфейс наблюдателя ядра можно прочитать в разделе 3.1.3.7 «Драйверы ядра».

3.1.3.8 Драйверы ядра

Понятие драйвера ядра введено в программную архитектуру службы мониторинга для разделения функциональных особенностей системы на составляющие. Это наделяет систему модульными особенностями - гибкостью и расширяемостью. Для расширения функционала конечной системы достаточно реализовать один или несколько дополнительных драйверов.

Драйверы реализуют функционал основных подсистем службы – транспортной и исполнительной. Можно условно разделить драйверы ядра на две группы – драйверы транспортной подсистемы и драйверы исполнительной подсистемы. Кроме того, существует третья группа драйверов, реализующих функционал самой платформы.

Драйверы транспортной подсистемы реализуют следующие функциональные особенности ядра:

- а) обнаружение в сети вычислительных узлов с запущенными службами мониторинга;
- б) управление удаленными сессиями;
- в) мониторинг доступности сетевой подсистемы узла.

Драйверы исполнительной подсистемы реализуют следующие особенности поведения ядра:

- а) планирование процесса запуска моделей мониторинга;
- б) запуск модулей мониторинга;
- в) получение и обработка результатов запуска модулей мониторинга;
- г) управление модулями мониторинга, развернутыми на данном узле.

В свою очередь, функциональные драйверы реализуют следующие особенности:

- а) получение системной информации о вычислительном узле;
- б) конфигурирование службы мониторинга;
- в) удаленное управление службой мониторинга.

В общем случае, драйвер ядра представляет собой сущность, характеризующуюся:

- а) специфичным поведением, реализующим некоторую часть общего функционала системы;
- б) наличием самостоятельного потока исполнения;
- в) возможностью запуска/останова;
- г) возможностью активации/деактивации;
- д) возможностью загрузки/выгрузки из памяти платформы;
- е) возможностью реагировать на изменение ядром своего состояния.

Рассмотрим общий интерфейс драйвера службы мониторинга на рисунке 3.7.

```
package com.googlecode.snoopyd.driver;

import com.googlecode.snoopyd.core.Kernel;

public interface Driver {
    public Kernel kernel();
    public String name();
}
```

Рисунок 3.6 – Интерфейс драйвера ядра

Обобщение интерфейса драйвера ядра было сделано в первую очередь для однообразной работы ядра с любым типом драйверов. Кроме того, так называемая точка устойчивости [link], реализованная в виде интерфейса, дает возможность прозрачно наращивать функционал системы.

Любой драйвер системы должен реализовывать как минимум два публичных метода: получение имени драйвера и получение ядра службы. Эти данные необходимы для корректной регистрации драйвера в системе и его последующей инициализации. Стоит отметить, что благодаря обобщенному интерфейсу все драйвера инициализируются и загружаются автоматически.

Кроме того, существуют дополнительные интерфейсы драйверов, которые могут быть реализованы опционально:

- а) загружаемый драйвер (рисунок 3.7);
- б) активируемый драйвер (рисунок 3.8);
- в) запускаемый драйвер (рисунок 3.9).

Каждый из перечисленных интерфейсов наделяет драйвер дополнительным поведением, которое кратко рассмотрено ниже.

Драйвер является загружаемым, если для его нормального функционирования требуется выполнить дополнительный набор действий в процессе загрузки или выгрузки модуля. Фактически, любой модуль является автоматически загружаемым, однако не каждый модуль требует при этом переопределения поведения загрузки и выгрузки.

```
package com.googlecode.snoopyd.driver;

import com.googlecode.snoopyd.core.Kernel;

public interface Loadable {
    public void load();
    public void unload();
}
```

Рисунок 3.7 – Интерфейс загружаемого драйвера

Следующим этапом после загрузки драйвера является его активация. Активация драйверов ядра происходит после полной загрузки ядра, инициализации всех структур и активации подсистем. Если требуется переопределить поведение драйвера в момент активации, он должен реализовывать соответствующий интерфейс.

```
package com.googlecode.snoopyd.driver;

public interface Activable {
    public void activate();
    public void deactivate();
}
```

Рисунок 3.8 – Интерфейс активируемого драйвера

Наконец запускаемые драйверы характеризуются собственным потоком исполнения и могут быть запущены или остановлены в любой момент эксплуатации системы. Запуск и остановка драйверов тесно связаны с понятием наблюдателя ядра. Как правило, драйверы останавливаются и запускаются только при переходе ядра в определенное ожидаемое состояние.

```
package com.googlecode.snoopyd.driver;

public interface Startable {
    public void start();
    public void stop();
    public boolean started();
    public void restart();
}
```

Рисунок 3.9 - Интерфейс запускаемого драйвера

В текущей реализации системы присутствует полный, по мнению авторов, набор драйверов, обеспечивающих систему полным функционалом.

Диаграмма классов реализованных не текущий момент в системе драйверов ядра приведена на рисунке 3.10.

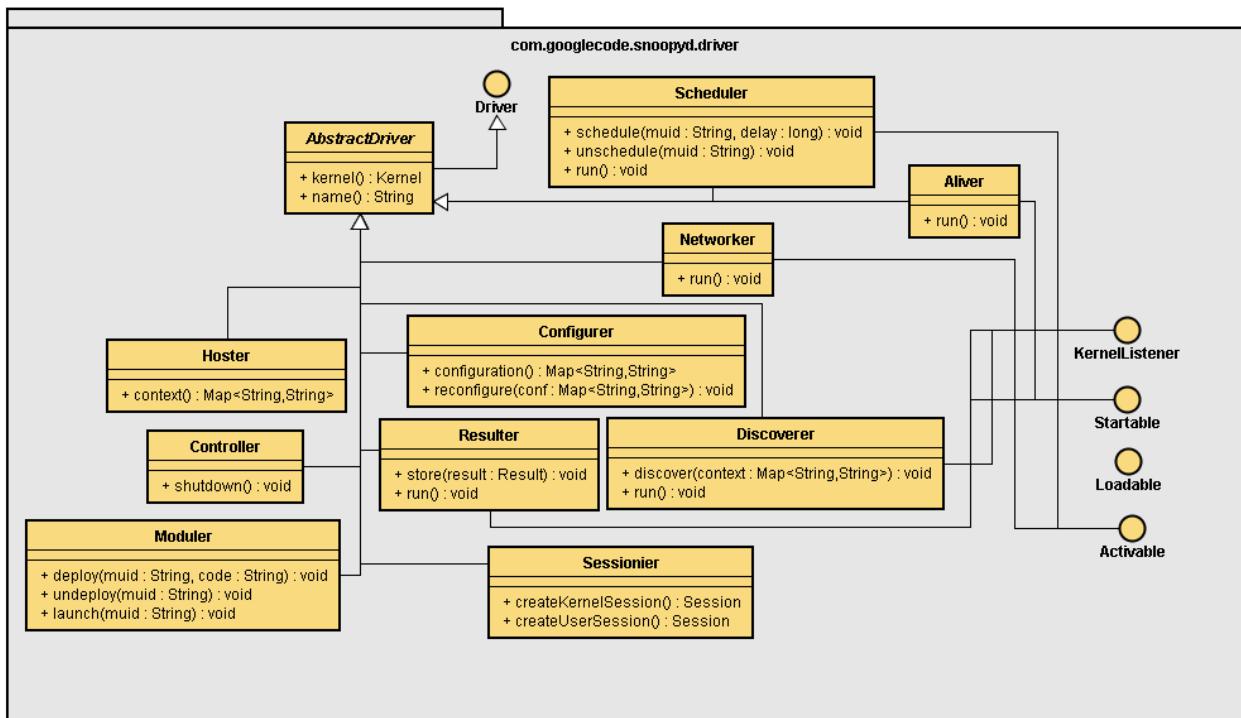


Рисунок 3.10 – Диаграмма классов драйверов

Краткое описание реализованных драйверов приведено в таблице 3.1.

Таблица 3.1 – Драйверы ядра

Драйвер	Интерфейсы	Основные функции
«Scheduler»	«наблюдатель»; «запускаемый»; «активируемый»;	планирование запусков модулей; управление персональным расписанием; синхронизация удаленных расписаний;
«Resulter»	«наблюдатель»; «запускаемый»; «активируемый»;	получение результатов выполнения модулей мониторинга; обработка результатов; пересылка результатов в постоянное хранилище;
«Networker»	«активируемый»;	мониторинг сетевой подсистемы узла; оповещение ядра о сбоях в работе сети;
«Aliver»	«запускаемый»; «наблюдатель»;	мониторинг доступности удаленных сессий ядра;

Продолжение таблицы 3.1

Драйвер	Интерфейсы	Основные функции
«Configurer»	«загружаемый»	конфигурирование свойств ядра;
«Discoverer»	«запускаемый»; «наблюдатель»;	обнаружение удаленных служб в сети; генерация событий обновления кеша ядра;
«Hoster»	«загружаемый»;	получение информации о вычислительном узле;
«Controller»	«загружаемый»;	удаленно управлением поведением ядра;
«Sessionier»	«загружаемый»;	создание сессий режима ядра и режима пользователя; управление удаленными сессиями;
«Moduler»	«загружаемый»;	запуск модулей мониторинга; развертывание модулей мониторинга; управление доступными модулями;

3.1.3.9 АдAPTERЫ ЯДРА

Для взаимодействия между драйверами, загруженными в различные адресные пространства, применяются специальные Ice-объекты - адAPTERЫ ЯДРА. При этом для использования удаленного драйвера достаточно в локальном адресном пространстве создать для его адAPTERА Ice-прокси. Такая связь называется «объект-прокси» и присутствует в архитектуре службы в качестве списков дочерних и родительских узлов (раздел 3.1.3.2 «Архитектура ядра»).

Рассмотрим на примере реализацию интерфейса адAPTERА драйвера Discoverer на языке описания спецификаций Slice[link] (рисунок 3.11). Более подробно про семантику работы драйвера Discoverer можно прочитать в разделе 3.1.3.2 «ИСследователь».

```
module com.googlecode.snoopyd.driver
{
    interface IDiscoverer
    {
        void discover(Ice::Identity identity);
    };
}
```

Рисунок 3.11 – Пример реализации интерфейса адаптера ядра

Можно условно разделить адаптеры ядра на две группы:

- а) стационарные адаптеры;
- б) динамические адаптеры;

Стационарные адаптеры создаются при запуске системы и доступны по уникальному статическому идентификатору на всем протяжении работы приложения. Это сделано для того, чтобы независимо от текущего состояния ядра и распределенной системы в целом иметь доступ к критическим частям службы мониторинга.

В текущей реализации доступно два стационарных адаптера – адаптеры для драйверов Discoverer и Sessionier (раздел 3.1.3.7 «Драйверы ядра»).

Динамические адаптеры генерируются автоматически на основании текущих подключённых к ядру сессий. Динамические адаптеры могут быть доступны только в рамках сессий – сессии режима ядра или сессии режима пользователя (раздел 3.1.3.10 «Сессии ядра»).

3.1.3.10 События ядра

События ядра наряду с его состояниями являются основополагающими компонентами всей программной платформы службы мониторинга. Внедрение в программную архитектуру модели генерации и обработки событий позволило исключить прямые взаимодействия между драйверами системы, тем самым придерживаются шаблоны «Низкая связность» и «Высокое зацепление» [link] в процессе разработки.

Событие ядра представляет собой сущность, характеризующуюся следующим:

- а) типом случившейся ситуации;
- б) списком параметров;
- в) наличием соответствующего обработчика.

С точки зрения реализации событие ядра представляется интерфейсом, представленным на рисунке 3.12.

```
package com.googlecode.snoopyd.core.event;

public interface KernelEvent {
    public String name();
}
```

Рисунок 3.12 – Интерфейс события ядра

Любое событие может быть сгенерировано только двумя сущностями – самим ядром, либо одним из его драйверов. При этом источник события заносит всю необходимую информацию в структуру и передает ее ядру на обработку, вызывая метод ядра из публичного интерфейса.

Диаграмма классов событий ядра, реализованных в текущей версии системы, представлена на рисунке 3.13.

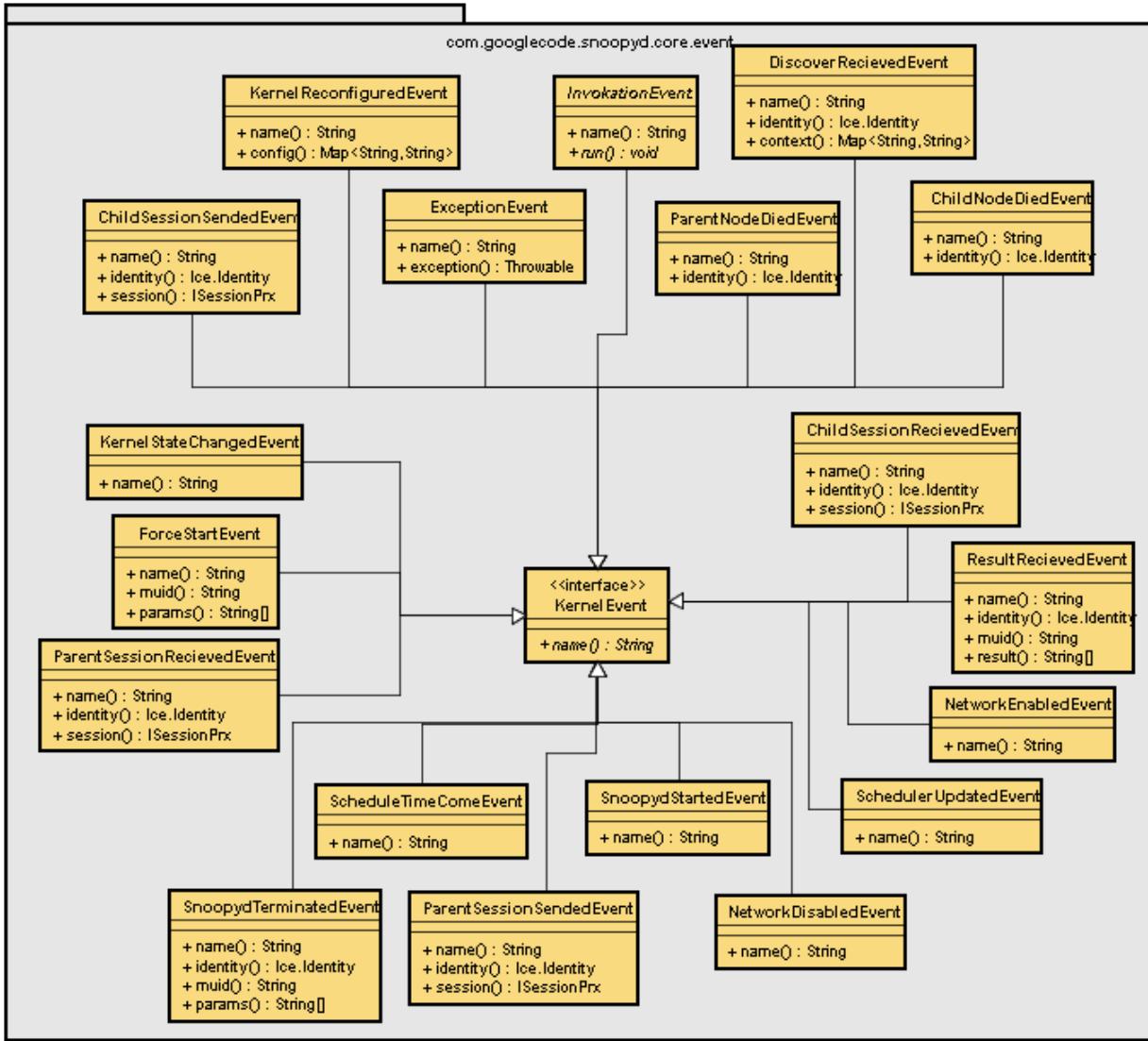


Рисунок 3.13 – Диаграмма классов событий ядра

События инкапсулируют тип возникшей ситуации в своем классе. Иными словами, идентификация типа события происходит на уровне поддержки полиморфизма языковой платформой. Такая реализация с точки зрения авторов является наиболее эффективной и расширяемой.

В таблице 3.2 кратко рассмотрены все события, поддерживаемые системой в текущей реализации.

Таблица 3.2 – События ядра

Событие	Источник	Описание
InvokationEvent	Scheduler	Абстрактное событие для инкапсулирования запуска модуля мониторинга, для которого сработало расписание (раздел 3.1.4.2 «Планировщик подсистемы исполнения»);
KernelReconfiguredEvent	Configurer	Событие, генерируемое при переконфигурации ядра;
ExceptionEvent	<any driver>	Событие, инкапсулирующее внешнее исключение драйвера, передаваемое на обработку ядру;
ChildSessionSendedEvent	Kernel	Событие, генерируемое ядром при посылке собственной сессии внешнему клиенту в качестве дочерней;
ChildSessionRecievedEvent	Kernel	Событие, генерируемое ядром при получении удаленной дочерней сессии узла;
ParentSessionSendedEvent	Kernel	Событие, генерируемое ядром при посылке собственной сессии внешнему клиенту в качестве родительской;
ParentSessionReceivedEvent	Kernel	Событие, генерируемое ядром при получении удаленной родительской сессии узла;
KernelStateChangedEvent	Kernel	Событие, генерируемое ядром при изменении своего состояния;

Продолжение таблицы 3.2

Событие	Источник	Описание
ForceStartEvent	Moduler	Событие, генерируемое при принудительном запуске модуля;
ChildNodeDiedEvent	Aliver	Событие, генерируемое при обнаружении «мертвой» дочерней сессии;
ParentNodeDiedEvent	Aliver	Событие, генерируемое при обнаружении «мертвой» родительской сессии;
DiscoverRecievedEvent	Discoverer	Событие, генерируемое при получении пакета обнаружения узлов;
ResultRecievedEvent	Scheduler	Событие, генерируемое при получении результат исполнения модуля мониторинга;
NetworkEnabledEvent	Networker	Событие, генерируемое при отсутствии сетевой подсистемы узла;
NetworkDisabledEvent	Networker	Событие, генерируемое при доступности сетевой подсистемы узла;
SchedulerUpdatedEvent	Scheduler	Событие, генерируемое планировщиком, при изменении внутреннего расписания службы;
ScheduleTimeComeEvent	Scheduler	Событие, генерируемое планировщиком, при наступлении времени исполнения модуля;
SnoopydStartedEvent	Kernel	Событие, генерируемое ядром при запуске службы;
SnoopydTerminatedEvent	Kernel	Событие, генерируемое ядром при завершении работы службы;

С точки зрения авторов, рассмотренный выше набор событий является абсолютно полными и позволяет описать любой сценарий использования.

3.1.3.11 Сессии ядра

Для удаленного взаимодействия между службами мониторинга нами были спроектированы и реализованы так называемые сессии ядра. Сессия ядра представляет собой Ice-прокси удаленного ядра и характеризуется:

- а) типом или классом сессии;
- б) методом вызовов удалённых процедур;
- в) списком публичных драйверов, методы которых можно вызывать удаленно;
- г) параметрами сетевого соединения (протокол, адрес, шифрование).

Можно выделить два вида сессий, доступных в текущей реализации: сессии режима ядра и сессии режима пользователя (рисунок 3.14).

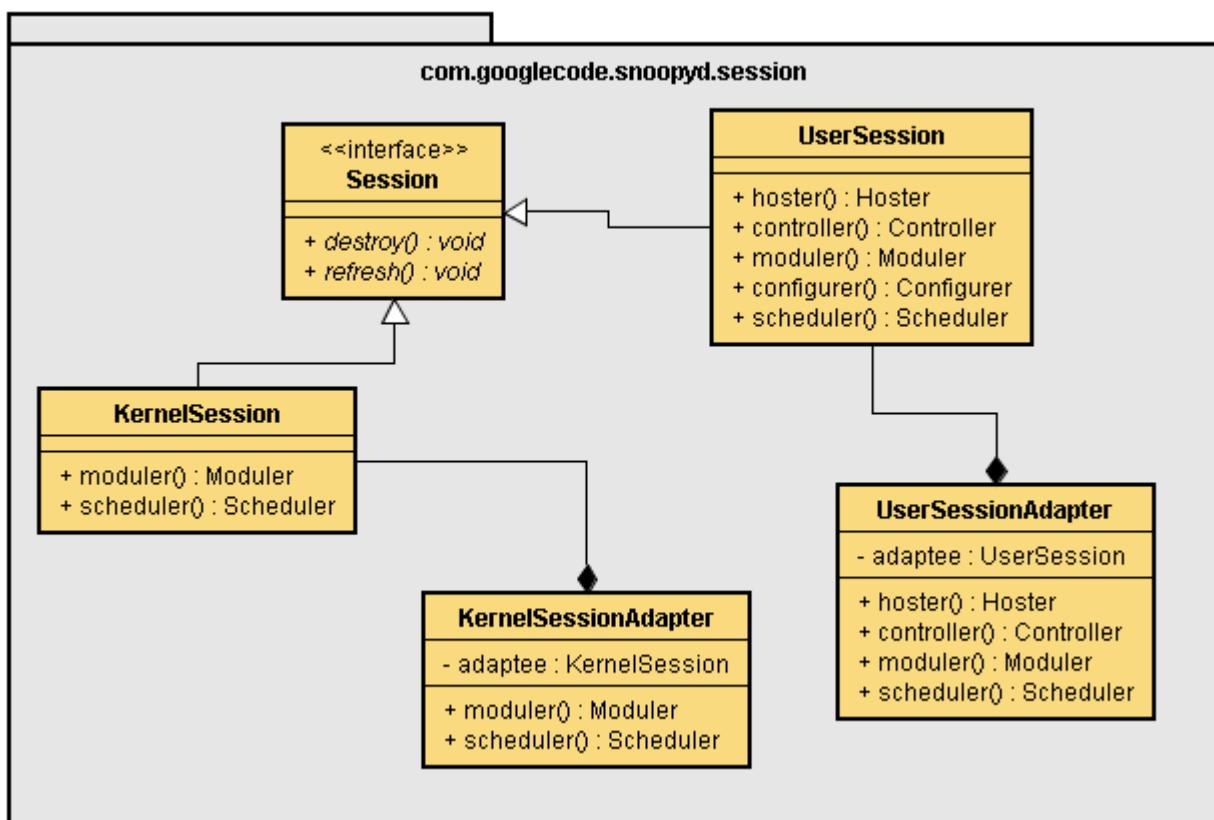


Рисунок 3.14 –Диаграмма классов сессий ядра

Сессии режима ядра устанавливаются между двумя удаленными драйверами, в то время как сессии режима пользователя устанавливаются между ядром и панелью управления.

Исходный код описаний интерфейсов на языке Slice для сессии ядра приведен на рисунке 3.15.

```
module session
{
    interface ISession
    {
        void refresh();
        void destroy();
    };

    interface IKernelSession extends ISession
    {
        driver::IModuler* moduler();
        driver::IScheduler* scheduler();
    };

    interface IUserSession extends ISession
    {
        driver::IHoster* hoster();
        driver::IController* controller();
        driver::IModuler* moduler();
        driver::IConfigurer* configurer();
        driver::IScheduler* scheduler();
    };
}
```

Рисунок 3.15 – Исходный код спецификаций сессий ядра

3.1.3.12 Индекс производительности узла

Для численной оценки вычислительных ресурсов узла, на котором запущена служба мониторинга, авторами была разработана шкала оценки ресурсов вычислительной системы.

Основная идея предлагаемой шкалы заключается в нормировании результатов оценки по некоторому базовому значению. В данном случае мы использовали абсолютный показатель производительности системы с

процессором Intel™ CoreDuo™ T2300 (1.66 ГГц), доступной оперативной памятью 1 Гб и свободным дисковым пространством 80 Гб. Абсолютная оценка для этой системы составляет 812.

Абсолютная оценка производительности узла является эвристической величиной и рассчитывается по основным ресурсам вычислительной системы – процессору, памяти, доступному дисковому пространству и текущей загруженности системы.

Любая оценка производительности узла нормируется на это значение. Фактически индекс производительности отражает во сколько раз исследуемая система производительней базовой.

Аналогичный подход используется в тестах оценки производительности компиляторов, виртуальных машин и баз данных SPEC[link].

3.1.3.13 Поведение ядра

При запуске ядро системы находится в «неопределенном состоянии», это длится до тех пор, пока ядро не получит одно из двух типов событий – «сеть доступна» или «сеть не доступна». Эти события переводят ядро в сетевое и автономное состояния соответственно.

Находясь в сетевом состоянии, ядро запускает драйвер Discoverer, который начинает «исследовать» среду на предмет наличия запущенных узлов.

После получения определенного количества пакетов типа Discoverer ядро принимает решение о подключении к какому-либо удаленному ядру. Выбор наиболее подходящего ядра осуществляется по принципу максимального индекса производительности. Эта информация доступна через контекст ядра.

Под подключением в данном контексте понимается обмен удаленными сессиями ядер. При этом одно ядро/узел становится родительским, другое – дочерним.

После осуществления обмена сессиями ядро переходит в следующее возможное состояние – активное или пассивное. Если ядро является родительским, его состояние становится активным, иначе – пассивным.

Автономное состояние ядра характеризуется отсутствием физического соединения с сетевой средой. В этом режиме ядро функционирует с некоторыми ограничениями. Однако при обнаружении сетевого соединения ядро переходит в сетевое состояние.

Активное, пассивное или автономное состояние ядра считаются заключительными и пригодными для нормальной эксплуатации. В то время как сетевое и неопределенное состояния являются промежуточными, своего рода искусственными состояниями.

3.1.4 Транспортная подсистема

3.1.4.1 Общее описание

Транспортная подсистема (рисунок 3.16) службы мониторинга реализует основную часть механизмов и примитивов модели распределенного взаимодействия между узлами. Транспортная подсистема представляет собой совокупность следующих логических компонент:

- а) ядра;
- б) драйверов транспортного уровня;
- в) менеджера сессий;
- г) многопоточных распределенных алгоритмов.

Транспортная подсистема реализует следующий функционал службы мониторинга:

- а) управление удаленными сессиями;
- б) мониторинг сетевой активности;
- в) именование объектов;
- г) адресация;
- д) балансировка нагрузки;
- е) выбор лидера.

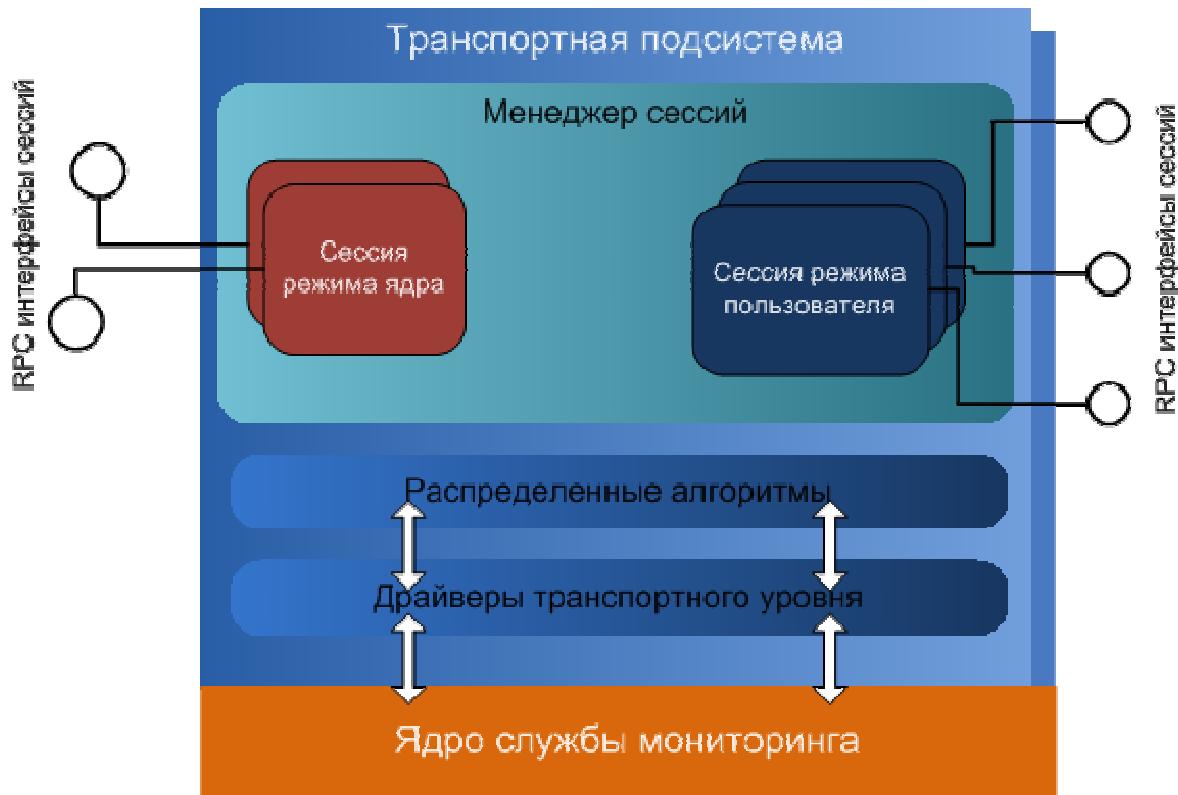


Рисунок 3.16 – Транспортная подсистема

3.1.4.2 Универсальный уникальный идентификатор (UUID)

Для однозначной идентификации объектов в рамках системы авторами использовалось понятие универсального уникального идентификатора (UUID). UUID представляет собой строку из 36-ти символов в формате Unicode (например такую — «5029a22c-e333-4f87-86b1-cd5e0fcce509»).

Стандартная библиотека классов Java уже содержит реализацию UUID в классе `java.util.UUID`.

Понятие универсального уникального идентификатора введено для однозначной идентификации узлов распределенной системы, сообщений, модулей мониторинга, а также любых других сущностей, требующих идентификации в распределенной системе.

Использование строкового идентификатора в данном случае оправдано по нескольким причинам. Во-первых, производительность современных вычислительных систем настолько высока, что они одинаково быстро работают как со строками, так и числовыми данным. Однако использование подобного подхода удовлетворяет требованиям к масштабируемости по отношению к

размеру системы. Во-вторых, согласно используемой платформе среднего слоя, все данные, передаваемые по каналам связи, упаковываются в бинарные последовательности и сжимаются, что позволяет снизить объемы сетевого трафика.

3.1.4.3 Уникальный идентификатор узла

Для эффективной работы транспортного уровня системы каждый узел идентифицируется не локальным или физическим адресом, а так называемым уникальным идентификатором узла (NUID). Идентификатор NUID состоит из двух частей — домена и идентификатора в домене. Под доменом распределенной системы мониторинга здесь и далее будем понимать объединенную группу узлов, с запущенными службами мониторинга, способными без каких-либо ограничений взаимодействовать между собой. В некотором смысле домен распределенной системы можно представлять как домен Windows, а узлы распределенной системы как компьютеры в домене [link].

Уникальный идентификатор узла позволяет избежать коллизий на транспортном уровне, разрывает связь между логическим узлом и его реальным адресом в сети.

3.1.4.4 Алгоритм выбора лидера

За основу алгоритма выбора лидера в предлагаемой архитектуре был взят классический алгоритм Чанди-Робертса [link], который основывается на кольцевой топологии сети с односторонней передачей данных. На его базе нами был разработан алгоритм выбора лидера, основанный на широковещательных запросах.

На практике распределенная система всегда образует полный мультиграф. Это необходимо для поддержания таких свойств распределенной системы, как репликация, переносимость и масштабируемость. Построение мультиграфа, достигается за счет посылки специальных служебных сообщений от каждого узла – каждому. В случае с использованием широковещательных протоколов эту операцию можно унифицировать.

Рассмотрим алгоритм выбора лидера:

- а) каждый узел отправляет широковещательный запрос в распределенную систему, содержащий свой индекс производительности;
- б) каждый узел получает широковещательные запросы от других узлов, содержащие их индексы производительности и сохраняет их в кеш;
- в) если новые узлы перестают обнаруживаться в среде или прошел определенный период времени, каждый узел начинает выбирать лидера из узлов, индексы которых записаны в кеше;
- г) выбор лидера представляет собой циклический алгоритм, состоящий из:
 - а. выбора узла из кеша (с удалением) с максимальным индексом производительности;
 - б. попытки подсоединения к выбранному узлу;
 - в. если попытка оказалась неудачной, из кеша будет взят следующий узел с максимальным индексом;
 - г. цикл повторяется до тех пор, пока состояние узла не изменится (пока лидер не будет выбран).

Алгоритм будет работать даже в случае присутствия одного узла в распределенной сети. Это обусловлено особенностями современных сетевых протоколов: широковещательный запрос получают все узлы сети, без исключения, независимо от отправителя запроса.

3.1.4.5 Обнаружение узлов

Каждый узел (ядро) характеризуется своим внутренним состоянием, которое состоит из:

- а) идентификатора узла;
- б) имени узла в сети;
- в) типом операционной системы, в которой запущен узел;
- г) строковыми представлениями адаптеров узла;

- д) индексом производительности узла;
- е) списком дочерних узлов (идентификаторов);
- ж) списком родительских узлов (идентификаторов).

С точки зрения авторов этот список является достаточно полным и может гарантировать любое изменение состояния распределенной системы после сбоев.

В данной реализации авторами был спроектирован и реализован механизм обнаружения соседних узлов. Он основан на использовании широковещательных запросов без гарантии доставки.

Для реализации подобного механизма нами был спроектирован специальный драйвер ядра и его стационарный адаптер – Discoverer. Discoverer – драйвер с самостоятельным потоком исполнения, который через определенный интервал посыпает в сеть широковещательный запрос, в котором инкапсулирует внутреннее состояние своего ядра.

Таким образом, в распределенной системе каждый отдельно взятый модуль имеет представление обо всей системе целиком и может быть использован для восстановления работоспособности системы после сбоев (репликация, масштабируемость). Схожие подходы применяются в современных методологиях разработки распределенных систем.

3.1.4.5 Сессии транспортной подсистемы

Сессии режима ядра и сессии режима пользователя (раздел «3.1.3.11 Сессии ядра») образуют множество сессий транспортной подсистемы, которые хранятся в так называемых таблицах дочерних и родительских узлов ядра службы.

Для обеспечения отказоустойчивости системы, а также для реализации поддержки механизмов автоматической балансировки нагрузки авторами был спроектирован специальный драйвер ядра – «Aliver» (раздел «3.1.3.8 Драйверы ядра»), который характеризуется самостоятельным потоком исполнения.

Для проверки доступности сессий авторами использовались существующие в платформе среднего слоя механизмы обмена мета-пакетами между распределенными объектами и прокси.

При обнаружении сессии, фактическое соединение через которую стало недоступным, драйвер «Aliver» генерирует ядру события «ParentNodeDied» или «ChildNodeDied», в зависимости от типа сессии. Обработка ядром этого события приводит либо к смене его внутреннего состояния (раздел «3.1.3.6 Состояния и обработчики ядра») либо к балансировке нагрузки.

3.1.4.7 Балансировка нагрузки

Любая распределенная система должна явно или неявно поддерживать механизмы балансировки нагрузки.

При явном подходе балансировка считается отдельным событием и случается либо через определенные периоды времени, либо по наступлению какого-либо другого события, сигнализирующего о том, что система в данный момент функционирует не достаточно эффективно.

При неявном подходе, система балансируется автоматически в процессе выбора лидера. Данный подход использовался авторами при реализации данной системы.

3.1.5 Подсистема исполнения

3.1.5.1 Общее описание

Подсистема исполнения службы мониторинга (рисунок 3.17) реализует основные механизмы исполнения модулей мониторинга. Можно выделить следующие компоненты подсистемы исполнения:

- а) ядра;
- б) драйверов подсистемы исполнения;
- в) менеджера модулей.

Менеджер модулей подсистемы исполнения представляет собой совокупность планировщика подсистемы исполнения и обработчика результатов.

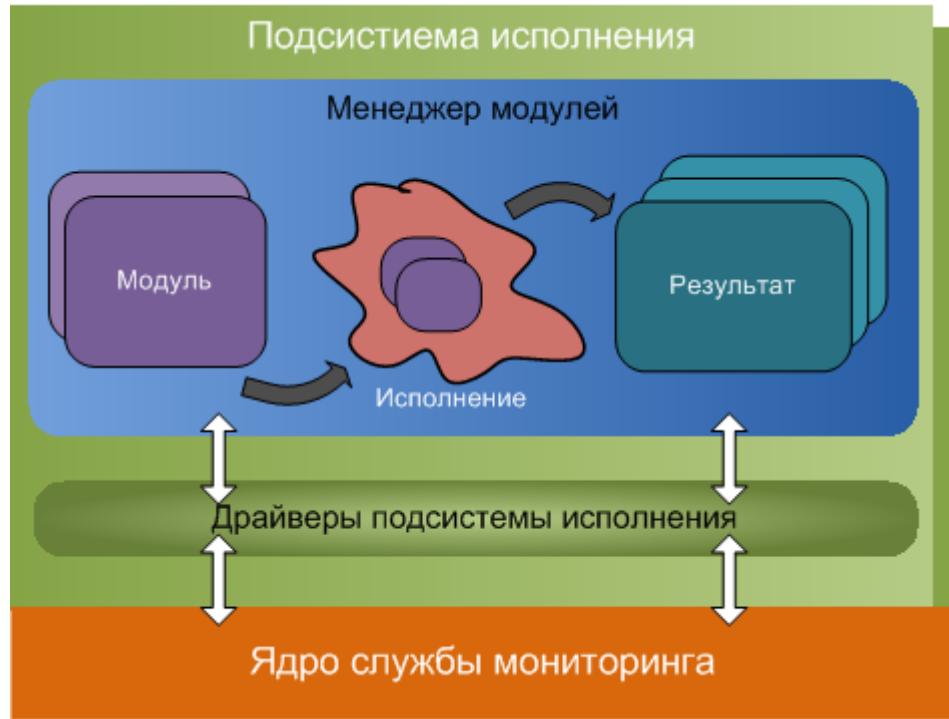


Рисунок 3.17 – Подсистема исполнения

Подсистема исполнения реализует следующий функционал службы мониторинга:

- а) планирование запусков и запуск модулей мониторинга;
- б) обработка результатов исполнения модулей;
- в) развертывание модулей мониторинга на удаленных узлах.

3.1.5.2 Расписание запусков модулей

В текущей реализации доступно два вида запусков модулей мониторинга:

- а) по расписанию;
- б) принудительно;

Расписание запусков модуля мониторинга представляет собой последовательность числовых интервалов запусков в секундах. На основании этой последовательности планировщик подсистемы исполнения осуществляет выполнение модулей мониторинга.

Кроме того, существует возможность принудительного запуска, который инициализируется пользователем через панель управления.

Расписание запусков модулей должно сохраняться в энергонезависимую память и иметь возможно восстанавливаться после сбоев или перезагрузок службы мониторинга.

3.1.5.3 Планировщик подсистемы исполнения

Планировщик подсистемы исполнения характеризуется самостоятельным программным потоком, который запускается при переходе ядра в активное состояние и останавливается при выходе из него.

Авторами было решено использовать модель делегирования обязанностей по планированию запусков модулей от дочерних узлов – родительским. Иными словами, планировщик, как сущность, присутствует исключительно на узлах, находящихся в активном состоянии и обслуживает одновременно все дочерние узлы.

Такой подход позволяет добиться эффективного использования распределенных ресурсов вычислительной среды.

3.1.5.4 Развёртывание модулей

Развёртывание модулей в распределенной системе осуществляется действиями пользователя через панель управления. При этом, на каждом узле будет сохранена локальная копия исходного текста модуля, ассоциированная с внутренним уникальным идентификатором (MUID).

Процесс развертываения осуществляется посредством широковещательных запросов в следующей последовательности:

- а) передача исходного текста модуля мониторинга;
- б) передача списка его параметров;
- в) передача расписания модуля.

Модули мониторинга сохраняются в локальном кеше узла именованные своими уникальными идентификаторами.

3.1.5.5 Обработчик результатов подсистемы исполнения

Полученные в результате запусков модулей результаты сохраняются в внутрисистемную очередь для последующей обработки.

Обработчик результатов, характеризующийся самостоятельным потоком исполнения, сохраняет результаты из очереди в ассоциированном с текущим узлом хранилище данных.

В текущей реализации службы мониторинга нет предварительной обработки и анализа результата.

3.2 Менеджер модулей

3.2.1 Общее описание

Менеджер модулей (рисунок 3.18) представляет собой обособленное приложение, взаимодействующее со службой мониторинга через удаленные RPC-сессии. Служба и соответствующей ей менеджер модулей должны быть запущены на одном узле распределенной системы мониторинга.

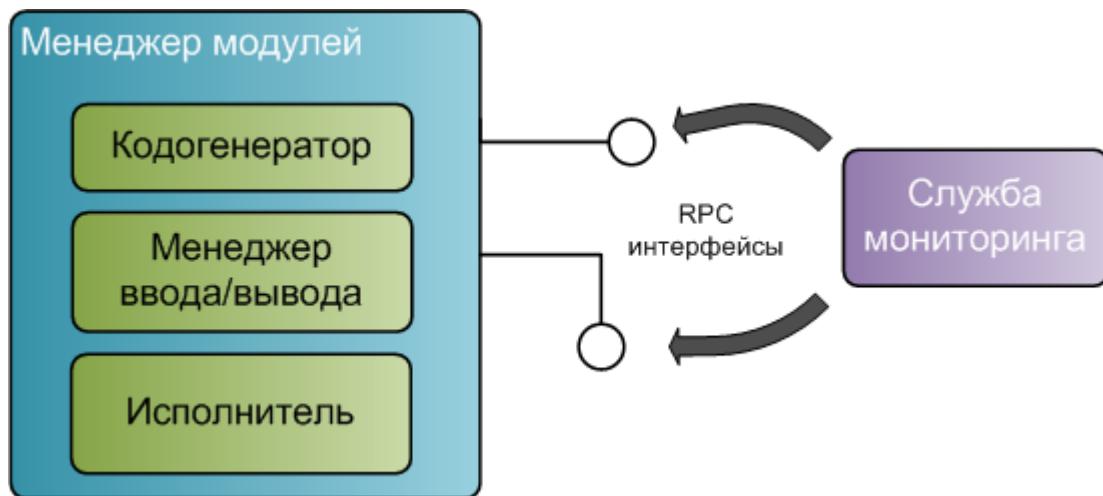


Рисунок 3.18 – Менеджер модулей

Менеджер модулей реализует следующий функционал распределенной системы мониторинга:

- генерация кода каркаса модулей;
- исполнение модулей мониторинга в операционной среде;

- в) выполнение низкоуровневых файловых операций при работе с модулями.

Авторами было решено разделить операции при работе с модулями на низкоуровневые и низкоуровневые. Например, развертываение модуля инициализируется службой мониторинга, подсистемой исполнения и обрабатывается как низкоуровневая файловая операция модулем мониторинга.

3.2.2 Выбор средств реализации

При выборе средств реализации менеджера модулей, авторы в первую очередь ориентировались на языки и средства выбранные для реализации интерфейса программирования модулей (раздел «3.3.3 Выбор средств реализации»). Это обусловлено возможностью динамического исполнения кода в интерпретируемых языках программирования, таких как Python и PHP.

При запуске модуля мониторинга генерируется так называемый код каркаса, который создает экземпляр модуля и запускает его. Очевидно, что реализовать такое поведение достаточно просто и прозрачно используя одни и те же средства как со стороны программного кода менеджера модулей так и со стороны кода самого модуля.

В итоге, авторами был выбран интерпретируемый язык общего назначения Python для реализации менеджера модулей.

3.2.3 Уникальный идентификатор модуля

Для идентификации модуля в рамках распределенной системы используется универсальный уникальный идентификатор (UUID) именуемый в дальнейшем уникальный идентификатор модуля (MUID). Процедура развертывания модуля на узле, помимо непосредственного сохранения модуля в памяти узла, подразумевает генерацию его уникального идентификатора.

3.3 Прикладной интерфейс программирования

3.3.1 Общее описание

Прикладной интерфейс программирования позволяет разрабатывать модули мониторинга на основе унифицированного каркаса исходного кода модуля. В текущей реализации интерфейс программирования модулей представляется каркасом с одним публичным методом – «`invoke(..)`» (рисунок 3.19). Параметры модуля мониторинга могут передаваться как обычная коллекция или список языка Python.

```
1  class Module(object):
2
3      def invoke(self, params):
4          raise NotImplementedError();
5
```

Рисунок 3.19 – Прикладной интерфейс программирования

Интерфейс программирования задает правила исполнения, передачи параметров и возврата результата модуля.

3.3.2 Выбор средств реализации

Среди поддерживаемых платформой Ice динамических языков программирования можно выделить два наиболее популярных – Python и PHP, поэтому при выборе языка программирования для интерфейса разработки модулей мониторинга, авторы рассматривали только эти два варианта.

В конечном счете, нами был выбрана платформа языка Python, как наиболее подходящая для реализации механизмов динамического расширения функционала. Язык Python обладает следующими преимуществами по отношению к PHP:

- а) более читабельный код и прозрачный синтаксис;
- б) ориентированность на различные задачи (не только WEB);
- в) большая библиотека стандартных модулей и классов;

- г) широкое распространение в сфере администрирования и автоматизации рутинных процессов;
- д) наличие и доступность широкого класса сторонних библиотек для решения круга повседневных задач.

3.4 Панель управления

3.4.1 Общее описание

Панель управления – инструмент управления работой исполняемой среды и визуализации информации процесса и результатов этой работы.

Панель помогает организовать процесс работы, а если более точно этот инструмент является интерфейсом между пользователем и исполняемой средой (рисунок 3.20). Через этот интерфейс задания доставляются от пользователя к узлу, на котором это задание должно быть выполнено и через него же возвращаются результаты работы.

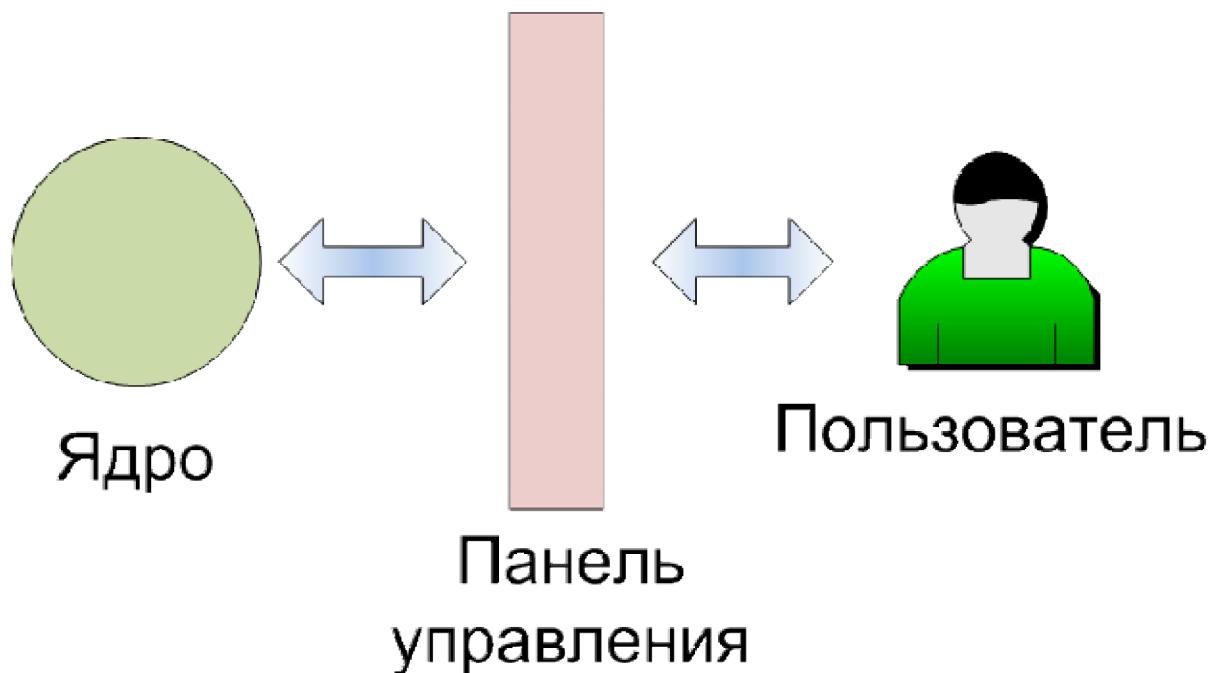


Рисунок 3.20 – Взаимодействие пользователя с ядром

Само приложение обладает большой мобильностью, вследствие отсутствия необходимости установки на жесткий диск компьютера и того, что оно написано на кроссплатформенном языке программирования Java. Конечно, для

функционирования программы требует, чтобы на компьютере должна быть установлена виртуальная машина Java.

Точкой входа панели в среду может быть любой узел сети, в которой функционирует исполняемая среда. При этом узел, в подавляющей большинстве это персональный компьютер, может быть даже не задействован в этой среде. Другими словами панель может быть запущена на любом компьютере в сети или подсети.

Приложение обладает простым интерфейсом содержащим элементы визуализации информации, такие как дерево узлов, граф домена, списки свойств и результатов, и элементы управления, такие как редактор модулей узла и средства удаленного развертывания модулей, элементы управления состоянием модулей и узлов.

3.4.2 Архитектура панели управления

Архитектура панели реализует такую концепцию как Модель – Представление – Поведение (рисунок 3.21). Суть этого шаблона проектирования заключается в том, что модель данных приложения, пользовательский интерфейс и управляющая логика разделены на три отдельных компонента так, что модификация одного из компонентов оказывает минимальное воздействие на остальные.

Важно отметить, что как представление, так и поведение зависят от модели. Однако модель не зависит ни от представления, ни от поведения. Это одно из ключевых достоинств подобного разделения. Оно позволяет строить модель независимо от визуального представления, а также создавать несколько различных представлений для одной модели.



Рисунок 3.21 – Концепция Модель-Представление-Поведение

3.4.3 Модель

Модель в терминах MVC — это не только совокупность кода доступа к данным, но и, как минимум, логика домена и, возможно, некоторые другие части системы.

3.4.3.1 Получение информации от ядра

Ядро информирует панель управления о своем существовании с помощью драйвера Discoverer. Это драйвер через интерфейс Discover() отправляет минимальный набор информации о ядре. Этот набор информации называется *контекстом*. Контекста достаточно чтобы зарегистрировать ядро в панели, а также для того, чтобы установить соединение с ядром. После того как соединение будет установлено, через панель можно будет получить более подробную справочную информацию о ядре и адаптеры для управления ядром.

Таблица 3.3 – Содержимое контекста ядра

Ключ для получения	Значение	Пример
identity	Идентификатор узла	dev/3f759634-59c0-42ae-a923-ad3e61eec769
os	Название операционной системы, на которой запущено ядро	Windows XP
parents	Идентификатор родительского узла	dev/92274582-5a96-47a9-942b-676119702ef6;
rate	Индекс производительности	17
primary	Информация для установления сессии до узла	tcp -h 192.168.0.10 -p 10000
secondary	Дополнительная информация для установления сессии до узла	udp -h 192.168.0.10 -p 10000
state	Состояние узла (статус)	ActiveState
hostname	Имя узла (компьютера) на котором запущено ядро	Work-PC
children	Идентификатор(ы) дочерних узлов	dev/9ea033b6-4a32-4b92-9a70-e57f56bb8d2c; dev/dcee7249-5baa-43c8-a1b2-2f787da599ec;

Авторами было принято решение, что такое количество информации является достаточным для актуального отображения узла в панели управления.

3.4.3.2 Взаимодействие с ядром системы

Использую информацию из контекста, устанавливается пользовательская сессия до удаленного узла. С помощью этой сессии получаются ссылки на драйверы удаленного ядра. Эти драйверы предоставляют интерфейсы, которые можно использовать для управления и конфигурирования узла.

Таблица 3.4 – Драйверы ядра используемые панелью управления

Драйвер	Интерфейсы/методы	Описание
Discoverer	discover()	Периодическое оповещение о состоянии удаленного узла через передаваемую информацию
Configurer	configuration()	Конфигурирование удаленного узла
Hoster	context()	Получение информации об удаленном узле. Отличается от контекста получаемого с помощью discover().
Sessionier	createUserSession()	Создание сессии до удаленного узла, которая впоследствии дает возможность получить ссылки на другие драйвера
Moduler	deploy()	Развертывание модуля на удаленном узле с передачей стартовых настроек
	force()	Принудительный старт модуля с возможностью передать параметры
	fetch()	Получение списка модулей узла
Scheduler	schedule()	Установления расписания для модуля удаленного узла
	toggle()	Переключение состояний модулей удаленного узла
	timetable()	Получения расписания модуля
	statetable()	Получения таблицы состояний модулей удаленного узла

Это основные методы для взаимодействия с ядром, но есть еще вспомогательные или, точнее, служебные. Например, такой метод как `ice_ping()` есть у всех драйверов. Задача его проста: определение достижимости удаленного узла. В случае недоступности узла метод генерирует исключительную ситуацию, на которую можно соответственно отреагировать.

3.4.3.3 Домен

В панели управления реализацией модели является класс `Domain`. Имеется еще несколько классов, которые являются вспомогательными для класса `Domain`. К ним можно отнести такие классы как `DomainController`, `Discoverer`, `DiscovererAdapter`, но они не являются ключевыми в реализации модели.

К функциям домена можно отнести:

1. Хранение контекста узлов ядра;
2. Хранение ссылок на драйвера удаленного узла;
3. Предоставляет информацию по требованию других компонентов приложения;
4. Информирует пользовательский интерфейс об изменениях информации, хранящейся в домене;
5. Контролирует актуальность информации и обновляет ее при необходимости или по требованию;
6. Осуществляет первичное взаимодействие с ядром через драйвер `Discoverer`.

Вся информация о ядре системы и отдельных узлах, которая визуализируется пользователю, хранится локально в ОЗУ на узле, где запущена панель управления. При этом она не сохраняется в ПЗУ, а в случае перезапуска панели запрашивается у ядра повторно.

Домен содержит адаптер, который предоставляет ядру интерфейс, через который контекст доставляется панели. Это происходит каждые несколько секунд, чтобы панель отображала актуальную информацию.

Полученная информация хранится в динамических массивах, списках, которые в свою очередь содержатся в контейнере `Domain` (рисунок 3.22).

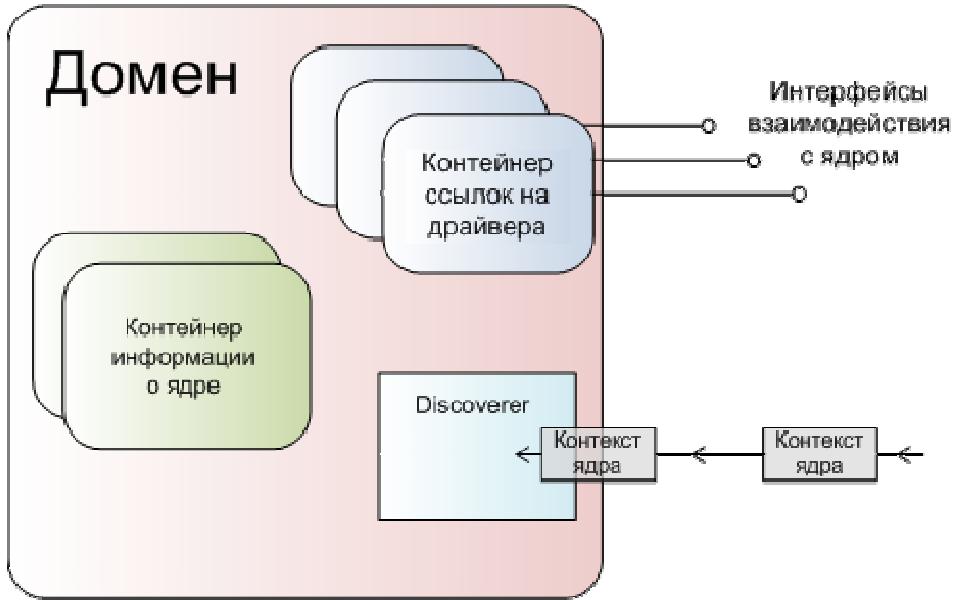


Рисунок 3.22 – Структура домена

Информация кэшируется и если она требуется, то берется из этого кэша, а не запрашивается у ядра каждый раз за исключением ситуаций, когда необходимо принудительно обновить информацию. Имеется два вида обновления информации. Первый вид - это ситуации, когда изменилось состояние ядра, а точнее его узлов. Тогда ядро через драйвер Discoverer оповещает об этом, рассылая новую информацию в контексте. Используя механизм хэшей, можно определить, изменилась ли информация, и если это произошло, то она обновляется в локальном кэше. Второй вид – ситуации, когда пользователь панели управления изменяет состояния ядра или отдельных его компонентов. И для того, чтобы узнать удачно ли прошли изменения, информация кэша также обновляется, но уже принудительным запросом. Примером второго вида является ситуация запроса пользователем результатов выполнения поставленного задания.

Такой подход позволяет снизить уровень трафика сетевого взаимодействия и повысить отзывчивость панели управления за счет того, что информация в большинстве случаев получается из локального хранилища, а не запрашивается постоянно у ядра.

Еще одним плюсом можно назвать возможность хранить информацию об узлах, которые неожиданно перестали отвечать (сообщать о своей активности). Если узел перестал отвечать, то он помечается как не отвечающий (dead), но

информация о нем не удаляется из кэша, а какое-то время хранится. Если до истечения определенного периода времени узел возобновил активность, то его восстанавливают в первоначальное состояние, при этом проверяется, не изменилась ли информация об узле и, если изменилась, то только тогда она обновляется. Если узел так и не возобновил активность, информация о нем удаляется из домена.

Для того чтобы в пользовательском интерфейсе отображалась актуальная информация, необходимо чтобы компонент ответственный за отрисовку обновлял ее своевременно. Постоянно проверять, не изменилась ли информация слишком расточительно с точки зрения ресурсов. И поэтому был использован поведенческий шаблон проектирования – Наблюдатель. Такое решение было принято по ряду требований к реализации:

- домен должен передавать сообщения интерфейсу, но при этом он не должен знать об его внутреннем устройстве;
- для предотвращения сильных связей между моделью и представлением.

Используемый шаблон полностью справляется с предъявленными требованиями.

3.4.4 Контроллер

Контроллер не должен содержать логики приложения (бизнес-логики). Таким образом Контроллер должен выполнять исключительно функцию связующего звена между отдельными компонентами системы.

3.4.4.1 Координатор

В панели управления реализацией контроллера или поведения является класс Coordinator.

Координатор:

1. Устанавливает соответствие между действиями пользователя и изменением информации в домене;

2. Согласовывает информацию из домена и ее отображение в пользовательском интерфейсе
3. Определяет основные действия, определяющие поведение панели в целом.

Первая заявленная функция координатора является основной.

3.4.5 Представление

В реализации представления нет какого-то конкретного класса. Оно реализуется целой группой классов, такие как окна приложения, модели таблиц, деревьев и вспомогательные классы. Но можно выделить основной класс, с которым взаимодействует координатор. Это класс главного окна приложения - MainFrame.

Поскольку панель управления написана на языке Java, то в качестве библиотеки для создания графического интерфейса использовался Swing. Swing предоставляет более гибкие интерфейсные компоненты, чем более ранняя библиотека AWT. Также выбранная библиотека является более кроссплатформенной.

3.4.5.1 Структура графического интерфейса

В качестве основы для графического интерфейса авторами была концепция многодокументного интерфейса (MDI). Это одна из стандартных архитектур пользовательского интерфейса Windows-приложений. MDI позволяет работать в приложении не с одним, а сразу с несколькими окнами или документами. Каждое окно показывается в пределах клиентской области основного окна приложения. Это правило не распространяется на модальные окна.

Такая организация интерфейса очень удобна, если необходимо работать с большим количеством однотипных окн. Подразумевается, что в панели управления будет открыто много окн, поэтому такая архитектура была выбрана, как наиболее подходящая.

3.4.5.2 Дерево узлов

Дерево узлов является основным средством отображения структуры ядра (рисунок 3.23). Так же через это дерево можно взаимодействовать с ядром, посредством выполнения над его узлами определенных действий, которые после обработки координатором будут отправлены выбранному узлу или даже группе узлов.

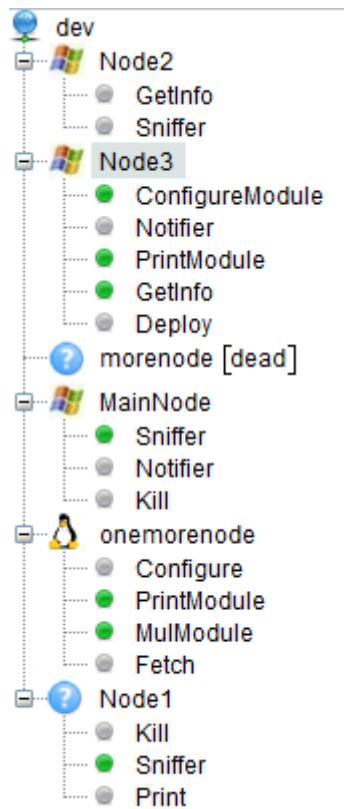


Рисунок 3.23 – Дерево узлов

В дереве можно встретить три вида узлов:

- домен;
- узел;
- модуль узла.

Как было сказано в разделе 3.1.3.2 узел в сети идентифицируется доменом и идентификатором (NUID). Первый вид узлов обозначает домены в сети. Как правило количество доменов в подсети редко превышает одного.

Ко второму виду относятся вычислительные узлы, которые были описаны в базовой теоретической модели в разделе 2.2. Это дерево не представляет

физической структуры узлов, потому что на одном программно-аппаратном устройстве, которое в большинстве случаев является персональный компьютер, можно запустить несколько сущностей ядра. Большая часть информации для дерева берется из контекста ядра, полученного через Discoverer. В качестве имени узла выбирается имя компьютера, а точнее то, что находится под значение hostname. Иконка для узла выбирается в зависимости от значения по ключу OS. Следует заметить, что круг устройств на котором может использоваться ядро не ограничивается персональными компьютерами, поэтому определение семейства операционной системы не всегда корректно, если вообще возможно. Если узел перестал информировать о своем существовании, то он помечается как dead. Поскольку узел недоступен, то и нет возможности получить список модулей.

У каждого узла может быть любое количество модулей, в том числе и ни одного. Список модулей можно получить через драйвер Moduler. Через интерфейс fetch() можно получить список состоящий из MUID и названия модуля. В дереве у каждого узла отображается список всех его модулей. У каждого модуля есть два состояния: активный и неактивный. Это состояние показывает индикатор модуля. Если индикатор зеленый, значит модуль активный, а если серый – неактивный. Статусы модулей получают через драйвер Scheduler через интерфейс statetable(), возвращающий список MUID и статус модуля.

С каждым узлом можно совершать определенный действия. Исключением является узел домена.

Действия доступные для узлов:

- Shutdown – прекратить выполнения ядра на удаленном узле;
- Host Results – вывести список результатов работы всех модулей узла;
- Node Properties – вывести свойства узла, позволяющие просмотреть системные свойства узла и провести конфигурацию ядра.

Действия доступные для модулей узла:

- Force Start – принудительный запуск модуля с параметрами;
- Module Results – вывести список результата работы конкретного модуля;

- Module Properties – вывести свойства модуля, которые содержат статус модуля, расписание и список параметров.

3.4.5.3 Карта узлов

Карта узлов представляет собой связанный граф. Для удобства карта открывается в отдельном окне (рисунок 3.24).

Важно понимать, что график отображает не физическую топологию сети и узлов. Здесь связи в графике показывают иерархическую связь, а не физическое соединение. Связь от узла MainNode к Node3 означает, что MainNode является родительским узлом по отношению к Node3. У любого узла должен быть родительский узел, поэтому MainNode связан сам с собой.

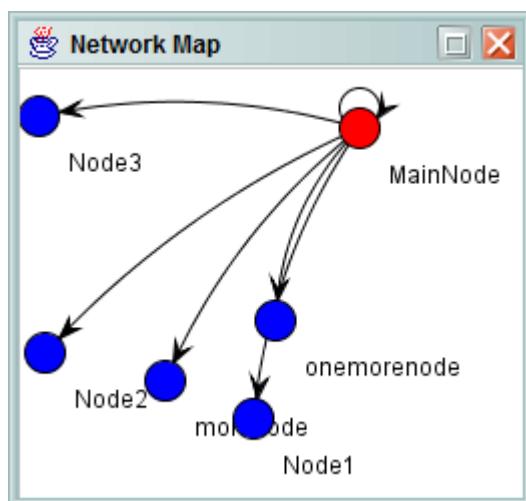


Рисунок 3.24 - Карта узлов с установленными родительскими связями

Цвет узлов на графике тоже имеет значение:

- красный – узел в активном состоянии;
- синий – в пассивном;
- желтый – в сетевом;
- серый – узел перестал отвечать (dead)

Первые три состояния полностью соответствуют состояниям ядра из раздела 3.1.3.5. Последнее же состояние неоднозначно. Если узел на графике окрашен в серый цвет, это не значит что ядро в автономном состоянии. Это значит, что текущее состояние ядро неизвестно. На самом деле оно может находиться в любом из возможных состояний.

Граф нарисован с помощью сторонней свободной библиотеки JUNG. Вся информация для отображения графа берется из контекста ядра.

3.4.5.4 Редактор

В панели управления имеется встроенный текстовый редактор. Он необходим для написания исходного кода модуля на языке Python. В редактор можно загрузить уже готовый файл и наоборот сохранить написанный. Из окна редактора можно сразу развернуть командой Deploy написанный или загруженный модуль на удаленный узел. Панель поддерживает множественно развертывание. Команда Deploy вызывает диалоговое окно, в котором предлагается из списка доступных узлов выбрать то или те, на которые необходимо развернуть модуль. В этом же окне сразу можно указать начальный статус модуля, параметры и расписание.

Развертывание модуля происходит через интерфейс `deploy()`, а выставление расписания и начальных параметров через `schedule()`.

3.4.5.6 Результаты выполнения задач

Результаты выполнения задач узел сохраняет в базе данных, адрес которой указан в его настройках. От туда же их получает панель управления но своими средствами, используя jdbc-драйвера.

Результаты для каждого узла или модуля извлекаются из базы согласно настройкам узла и отображаются в виде таблиц в отдельном окне для каждого узла и модуля.

3.5 Описание организации совместной работы

В виду того, что проект разрабатывался командой из двух человек, были приняты следующие решения по организации работы в группе разработчиков.

3.5.1 Skype (skype.com)

Из-за географической распределенности участников проекта, большинство митингов и коде ревью проходили в режиме «онлайн». Для организации видеоконференций мы использовали популярное приложение VoIP-телефонии Skype.

3.5.2 GoogleMail (gmail.com)

Для организации списка рассылки проекта использовались инструменты GoogleMail. Все технические моменты и решения обсуждались посредством переписки на английском языке в закрытом списке рассылки – snoopy@googlecode.com. С момента начала проекта было создано 18 почтовых веток общим объемом около 140-ти писем.

3.5.2 Хостинг проектов Google (goolecode.com)

Площадка для хостинга проектов от Google использовалась для организации репозитория исходного кода, баг-трекера и вики-движка. Проект доступен по ссылке – <http://snoopy.googlecode.com>.

В качестве системы хранения версий использовалась популярная централизованная система SVN. В репозитории хранились исходные тексты проекта, текст пояснительной записки, черновики схем и документов. С момента начала проекта было произведено около 250 ревизий исходного кода.

4 Организационно-экономический раздел

4.1 Расчет затрат на этапе проектирования

Для начала посчитаем трудоемкость программного продукта. Определим общие затраты труда Т по формуле:

$$T = T_o + T_i + T_a + T_p + T_{otl} + T_d, \quad (4.1)$$

где T_o – затраты труда на описание задачи;

T_i – затраты на исследование предметной области;

T_a – затраты на разработку блок-схем;

T_p – затраты на программирование;

T_{otl} – затраты на отладку;

T_d – затраты на подготовку документации.

Все составляющие определяем через условное число команд - Q:

$$Q = q \times c \times (1+p), \quad (4.2)$$

где q — предполагаемое число команд;

c — коэффициент сложности задачи;

p — коэффициент коррекции программы.

В результате оценки было получено предполагаемое число операторов, равное 6000.

Коэффициент сложности задачи характеризует относительную сложность программы по отношению к так называемой типовой задаче, реализующей стандартные методы решения, сложность которой принята равной единице (величина «с» лежит в пределах от 1,25 до 2). Для нашего программного продукта, включающего в себя алгоритмы распределенного сетевого взаимодействия, выполнения произвольных модулей сложность задачи возьмем 1,4.

Коэффициент коррекции программы характеризует увеличение объема работ за счет внесения изменений в алгоритм или программу по результатам уточнения постановок. С учетом того, что в нашем случае постоянно находились лучшие и эффективные алгоритмы взамен уже написанным, что приводило к частой замене кода, возьмем коэффициент равным 0,2.

В результате, согласно формуле (4.2) получим $Q = 6000 \times 1,4 \times (1 + 0,2) = 10080$ условное число команд.

Также используем следующие коэффициенты.

Коэффициент увеличения затрат труда, вследствие недостаточного описания задачи, в зависимости от сложности задачи принимается от 1,2 до 1,5, в связи с тем, что данная задача, потребовала уточнения и больших доработок, примем $B = 1,4$.

Коэффициент квалификации разработчика k определяется в зависимости от стажа работы и составляет: для работающих до двух лет – 0,8; от двух до трех лет - 1,0; от трех до пяти лет - 1,1 - 1,2; от пяти до семи - 1,3 - 1,4; свыше семи лет 1,5 — 1,6.

В нашем случае разработчики, которым было поручено это задание, имели опыт работы менее года, поэтому примем $k = 0,8$.

Рассчитаем общую трудоемкость.

Затраты труда на подготовку описания задачи Точно определить невозможно, так как это связано с творческим характером работы. Примем $T_o = 80$ чел.-ч.

Затраты труда на изучение описания задачи T_i с учетом уточнения описания и квалификации программиста могут быть определены по формуле: $T_i = Q \times B \times k / 80$. Где Q – условное число команд, B – коэффициент увеличения затрат труда, вследствие недостаточного описания задачи, $T_i = 10080 \times 1,4 \times 0,8 / 80 = 141,12$ чел.-ч.

Затраты труда на разработку алгоритма решения задачи T_a рассчитывается по формуле:

$$T_a = Q \times k / 25$$

$$T_a = 10080 \times 0,8 / 25 = 322,56 \text{ чел.-ч.}$$

Затраты труда на составление программы по готовой блок-схеме T_p определяется по формуле:

$$T_p = Q \times k / 25$$

$$T_p = 10080 \times 0,8 / 25 = 322,56 \text{ чел.-ч}$$

Затраты труда на отладку программы на ЭВМ T_{otl} рассчитывается по следующей формуле:

$$T_{otl} = Q \times k / 5$$

$$T_{otl} = 10080 \times 0,8 / 5 = 1612,8 \text{ чел.-ч.}$$

Затраты труда на подготовку документации по задаче T_d определяются по формуле:

$$T_d = T_{dp} + T_{do},$$

где T_{dp} - затраты труда на подготовку материалов в рукописи;

T_{do} – затраты труда на редактирование, печать и оформление документации.

$$T_{dp} = Q \times k / 20$$

$$T_{dp} = 10080 \times 0,8 / 20 = 403,2 \text{ чел.-ч.}$$

$$T_{do} = 0,75 \times T_{dp}$$

$$T_{do} = 0,75 \times 403,2 = 302,4 \text{ чел.-ч.}$$

В итоге:

$$T_d = 403,2 + 302,4 = 705,6 \text{ чел.-ч.}$$

С учетом уровня языка программирования трудоемкость разработки программы может быть скорректирована следующим образом:

$$T_{kor} = T \times k_{kor}, \quad (4.3)$$

где k_{kor} – коэффициент изменения трудоемкости, берущийся из таблицы 4.1.

Таблица 4.1 – Коэффициенты трудоемкости языков программирования

Уровень языка программирования	Характеристики языка программирования	Коэффициент изменения трудоемкости
1	Покомандныйавтокод- ассемблер	1
2	Макроассемблер	0,95
3	Алгоритмическиеязыкивысокогоуровня	0,8 — 0,9
4	Алгоритмическиеязыкисверхвысокогоуровня	0,7 — 0,8

Выбранный для разработки язык Java относится к алгоритмическим языкам сверхвысокого уровня, с учетом этого примем $k_{\text{кор}} = 0,85$.

Подставив все полученные данные в формулу (1), получим полную трудоемкость разработки: $T = 80 + 141,12 + 322,56 + 322,56 + 1612,8 + 705,6 = 3184,64 \text{ чел.-ч.}$

С учетом корректировки из формулы (4.3) получим итоговую трудоемкость разработки: $T_{\text{кор}} = 0,85 \times 3184,64 = 2707 \text{ чел.-ч.}$

Наиболее распространенный и простой подход к оценке трудоемкости имеет следующий алгоритм.

Применяя метод экспертных оценок, наиболее часто используют эмпирическую формулу

$$t_{\text{ож}} = (2 * t_{\max} + 3 * t_{\min}) / 5$$

где $t_{\text{ож}}$ – ожидаемая длительность работы;

t_{\min} – минимальная длительность работы (этапа) по мнению эксперта;

t_{\max} – максимальная длительность работы (этапа) по мнению эксперта.

Пример расчета ожидаемой продолжительности работ приведен в таблице 4.2

Таблица 4.2 – Ожидаемая продолжительность работ

Наименование работ	Длительность работ (дней)		
	Минимум	Максимум	Ожидаемая
1. Разработка технического задания	2	4	3
2. Анализ технического задания и сбор данных	4	7	5

Продолжение таблицы 4.2

Наименование работ	Длительность работ (дней)		
	Минимум	Максимум	Ожидаемая
3. Составление алгоритма	7	14	10
4. Реализация алгоритма на языке программирования Java	6	16	10
5. Набор программы на ПЭВМ	10	14	12
6. Отладка программы и тестирование	10	18	13
7. Проведение экспериментов	8	15	11
8. Оформление пояснительной записки	5	10	7

Суммарная продолжительность работ на этапе проектирования составляет 71 день (46 на компьютере).

Капитальные затраты на этапе проектирования рассчитываются по формуле

$$K_n = Z_n + M_n + H_n, \quad (4.4)$$

где Z_n – заработка плата проектировщика на всем этапе проектирования;

M_n – затраты на использование ЭВМ на этапе проектирования;

H_n – накладные расходы на этапе проектирования.

$$Z_n = Z_d T_n \left(1 + \frac{a_c}{100}\right),$$

где Z_d – дневная заработка плата разработчика на этапе проектирования (определяется по практическим данным конкретной организации);

T_n – продолжительность работ на этапе проектирования;

a_c – выплаты в государственные внебюджетные фонды;

a_n – процент премий.

Тогда, в нашем случае

$$Z_n = 476 * 71 * (1 + 0,34) = 45286,64 \text{ руб}$$

Расходы на машинное время состоят из расходов за процессорное время (при работе в сети) и расходов за дисплейное время. Формула для расчетов затрат на использование ЭВМ на этапе проектирования имеет вид

$$M_n = c_n * t_n ,$$

где c_n и c_d – соответственно стоимость одного часа процессорного и дисплейного времени, руб.;

t_n и t_d – время, необходимое для решения задачи, соответственно процессорное и дисплейное.

Для нашего примера, так как программа разработана на ЭВМ Celeron 1,6 GHz, в процессорном времени необходимости нет, т.е. $c_n = 0$ и $t_n = 0$.

Для подсчета машинного времени определяем, что ЭВМ необходима на этапах программирования, отладки, тестирования. С учетом того, что в день ЭВМ работает 6 часов, получаем

$$t_d = 46 * 6 = 276 \text{ ч.}$$

Исходя из этого определим затраты, связанные с ЭВМ:

$$M_n = 16 * 276 = 4416 \text{ руб}$$

Накладные расходы составляют 120 % от заработной платы персонала, занятого эксплуатацией программы (разработчика), и вычисляются по формуле

$$H_n = Z_n * 120/100,$$

$$H_n = 45286,64 * 1,2 = 54343,968 \text{ руб.}$$

Теперь рассчитаем капитальные затраты на этапе проектирования K_n по формуле (4.4) и получим:

$$K_n = 45286,64 + 4416 + 54343,968 = 104046,608 \text{ руб.}$$

4.2 Выбор базы сравнения

В качестве аналога рассмотрим такой программный продукт как Cacti. Cacti — веб-приложение (набор скриптов) которое поможет Вам мониторить состояние вашего сервера, CISCO и всего что может отдавать данные по SNMP протоколу. Обладает хорошим и понятным web-интерфейсом, расширяемостью за счет написания модулей для дополнительного функционала, наличием готовых шаблонов для многообразного оборудования.

Рассмотрим сравнительную характеристику с аналогом в таблице 4.3.

Таблица 4.3 – Сравнительная характеристика

Параметры и характеристики	Ручное проектирование	Автоматизированное проектирование
Распределенность	Нет	Да
Быстродействие	Среднее	Высокое
Надежность и удобствоиспользования	Нет	Да
Гибкость	Нет	Да
Расширяемость	Нет	Да

Каждому i -му ($i=5$) выбранному показателю для сравнения определим коэффициент k_i (коэффициент его весомости (важности)). Для этого каждый показатель оценим с использованием 10-балльной шкалы.

Оценки характеристик K_i и их соответствующие весовые коэффициенты k_i сведены в таблице 4.4

Таблица 4.4 – Распределение весовых коэффициентов по характеристикам

Характеристика	Весовой коэффициент важности k_i	Новое изделие (н)		Изделие	Аналог (а)
		Оценка характеристики, K_i	Значимость, $k_i * K_i$		
Распределенность	0,20	9	1,8	5	1
Быстродействие	0,15	7	1,05	8	1,2
Надежность и удобствоиспользования	0,17	7	1,19	6	1,02
Гибкость	0,26	8	2,08	6	1,56
Расширяемость	0,22	7	1,54	7	1,54
Итого:	1	38	7,66	32	6,32

Из формулы видно, что $K = 38/32 = 1,1875$ (больше 1), значит разрабатываемый продукт лучше, чем аналог.

Данные для расчета приведены в таблице 4.5.

Таблица 4.5 – Исходные данные для расчета экономического эффекта

N	C_d	Z_d	R	a_c	H	P	$p_{пр}$	E_H
1 чел.	16 руб/ч	238 руб/день	21	34%	120%	15%	3%	0,15

Численность обслуживающего персонала (N).

Стоимость машинного времени (C_d) (определяется по практическим данным конкретной организации).

Дневная заработка плата проектировщика (Z_d) (определяется по практическим данным конкретной организации).

$$Z_d = Z / R = 5000 / 21 = 238 \text{ руб} ,$$

где Z – заработка плата проектировщика за месяц, которая составляет 5000 рублей;

R – среднее число рабочих дней в месяце;

a_c – выплаты в государственные внебюджетные фонды;

a_p – средний процент премий за год;

H – накладные расходы от заработной платы;

P – расчетная прибыль от продажи;

$P_{пр}$ – прочие расходы;

E_H – нормативный коэффициент.

4.3 Сравнительный анализ затрат в ходе эксплуатации программного продукта и аналога

В качестве критериев для сравнения характеристик программного обеспечения наиболее важными для данного программного продукта являются эксплуатационные расходы:

- содержание персонала, занятого работой с программой

- расходы на функционирование ЭВМ
- накладные расходы
- прочие расходы

Расходы по различным видам работающих определяются следующей формулой:

$$Z = \sum n_i * \bar{z}_i * k \left(1 + \frac{a_c}{100}\right),$$

где n_i – численность персонала i -го вида;

\bar{z}_i – среднегодовая зарплата работников i -го вида;

k – процент занятости обслуживающего персонала при работе с аналогом и обслуживающим персоналом.

До внедрения программы время использования ЭВМ составляло:

$$t_1 = 252 * 6 = 1512 \text{ ч/год}$$

Таким образом, после внедрения программы время использования ЭВМ составит:

$$t_2 = 126 * 6 = 756 \text{ ч/год}$$

Процент занятости ЭВМ составляет:

- до внедрения программы

$$K_a = t_1 / (252 * 6) = 1512 / 1512 = 1$$

- после внедрения программы

$$K_{np} = t_2 / (252 * 6) = 756 / 1512 = 0,5$$

Численность персонала составляет $n_i = 1$ человек. Найдем z_i из произведения

$$z_i = z_i * t_p = 6000 * 12 = 72000 \text{ (руб.)}$$

где t_p – период работы, который составляет 12 месяцев;

$$z_i = T * z_d / 12 = 252 * 756 / 12 = 15876 \text{ (руб.)}$$

k_i – процентное исполнение работы с учетом занятости пользователя.

Рассчитаем заработную плату с начислениями на одного работника:

$$Z_1 = 1 * 72000 * 1 * 1,34 = 96480 \text{ (руб.)}$$

$$Z_2 = 1 * 72000 * 0,5 * 1,34 = 48240 \text{ (руб.)}$$

Расходы на функционирование программы складываются из затрат на эксплуатационные принадлежности (бумага, краска для принтера и т.д.) и периферийного оборудования (принтер, плоттер, дискеты и т.д.).

В общем случае расходы на машинное время состоят из расходов на процессорное время (при работе с объектным и абсолютным кодом) и расходов на дисплейное время. Формула для расчетов имеет вид

$$M_n = c_n * t_n + c_d * t_d$$

где c_n и c_d – соответственно стоимость одного часа процессорного и дисплейного времени, руб.; t_n и t_d – время, необходимое для решения задачи, соответственно процессорное и дисплейное. Так как программа разработана на современной мощной ЭВМ, то в процессорном времени необходимости нет, т.е.

$$c_n=0, t_n=0$$

$$t_{d1} = F_{\text{эф.р}} * K_a, t_{d2} = F_{\text{эф.р}} * K_{\text{пр}}, F_{\text{эф.р}} = N * t$$

где $F_{\text{эф.р}}$ – эффективное рабочее время;

N – количество рабочих дней в году;

t – продолжительность рабочего дня.

До внедрения программы:

$$C_d = 16 \text{ руб/ч}$$

$$N = 21 * 12 = 252 \text{ (дн)}$$

$$F_{\text{эф.р}} = 252 * 6 = 1512 \text{ (ч)}$$

$$M_1 = 16 * 1512 * 1 = 24192 \text{ (руб.)}$$

После внедрения программы затраты на использование ЭВМ составили:

$$t_{d2} = 252 * 0,5 * 6 = 756 \text{ (ч)}$$

$$M_2 = 16 * 1512 * 0,5 = 12096 \text{ руб.}$$

Накладные расходы составляют 120 % от заработной платы персонала, занятого эксплуатацией программы, и вычисляются по формуле

$$H = Z * 120/100$$

$$H_{n1} = 96480 * 1,2 = 115776 \text{ руб.}$$

$$H_{n2} = 48240 * 1,2 = 57888 \text{ руб.}$$

Прочие расходы составляют 3 % от суммы всех эксплуатационных расходов.

До внедрения программы они составили:

$$P_{p1} = (Z_1 + M_1 + H_1) * 0,03$$

$$P_{np1} = (96480 + 24192 + 115776) * 0,03 = 7093,44 \text{ руб.}$$

После внедрения программы они составили:

$$P_{p2} = (Z_2 + M_2 + H_2) * 0,03$$

$$P_{p2} = (48240 + 12096 + 57888) * 0,03 = 3546,72 \text{ руб.}$$

Таким образом, эксплуатационные расходы составили:

до внедрения программы

$$P_1 = Z_1 + M_1 + H_1 + P_{np1}$$

$$P_1 = 96480 + 24192 + 115776 + 7093,44 = 243541,44 \text{ руб.}$$

после внедрения программы

$$P_2 = Z_2 + M_2 + H_2 + P_{np2}$$

$$P_2 = 48240 + 12096 + 57888 + 3546,72 = 121770,72 \text{ руб.}$$

4.4 Расчет экономии от увеличения производительности труда пользователя

Если пользователь при выполнении работы J-го типа с применением программы экономит ΔT_j часов, то повышение производительности труда P_j (%) определяется по формуле

$$P_j = \frac{\Delta T_j}{F_j * \Delta T_j} * 100\% ,$$

где ΔT_j – экономия машинного времени при использовании разработанной программы (ч);

F_j – время, которое отводится пользователю для выполнения работы j -го типа до внедрения разработанной программы (ч).

$$\Delta T_j = t_{d1} - t_{d2}$$

$$\Delta T_j = 756 \text{ ч/год}$$

Тогда

$$P = 756 / (1512 - 756) * 100 \% = 100 \%$$

Экономия, связанная с повышением производительности труда пользователя ΔP , определяется по формуле

$$\Delta P_n = Z_n \sum_i \frac{P_i}{100}$$

где Z – среднегодовая заработка плата пользователя, с учетом процента занятости при использовании разрабатываемого проекта, равная

$$\Delta P_n = 48240 * 100/100 = 48240 \text{ руб}$$

Если программы используют пользователи разной квалификации, то расчеты следует выполнить отдельно по каждой k -й квалификации, при этом

$$P_j = \sum (\Delta P_n) k ,$$

где $(\Delta P_n)k$ – экономия, полученная от повышения производительности труда пользователя k -й квалификации.

4.5 Ожидаемый экономический эффект и срок окупаемости капитальных затрат

Ожидаемый экономический эффект рассчитывается после завершения предпроизводственной стадии создания программного продукта и определяется по формуле

$$\mathcal{E} = \mathcal{E}_d - K_h E_h$$

где \mathcal{E}_d – годовая экономия;

K_h – капитальные затраты на проектирование;

$$(E_h = 0,15)$$

E_h – нормативный коэффициент эффективности.

Годовая экономия \mathcal{E}_d складывается из экономии эксплуатационных расходов и экономии в связи с повышением производительности труда проектировщика:

$$\mathcal{E}_d = (P_1 - P_2) + \Delta P$$

где P_1 , P_2 – соответственно эксплуатационные расходы до и после внедрения программы;

ΔP_n – экономия от повышения производительности труда пользователя,

$$\mathcal{E}_d = (243541,44 - 121770,72) + 48240 = 170010,72 \text{ руб.}$$

Тогда ожидаемый экономический эффект составит:

$$\mathcal{E} = 170010,72 - 0,15 * 104046,608 = 154403,7288 \text{ руб.}$$

Рассчитаем цену реализации разработанного программного продукта, при условии, что планируемая прибыль от продажи должна составлять не менее 15 %.

Цену программного продукта рассчитаем по формуле

$$Ц = K_p(1 + П/100),$$

где $П$ – расчетная прибыль от реализации ($П=15\%$)

$$Ц = 104046,608 * (1 + 0,15) = 119653,5992 \text{ руб.}$$

Срок окупаемости капитальных затрат на проектирование программного продукта рассчитывается по формуле:

$$T = K_p/\mathcal{E}_d$$

$$T = 104046,608 / 170010,72 = 0,61 \text{ (год)} = 7,3 \text{ (мес)}$$

Полученный результат характеризует быстрый срок окупаемости затрат.

Данные расчетов экономических показателей сведены в таблицу 4.6.

Таблица 4.6 – Итоговая таблица

Экономические показатели	Буквенное обозначение	Единицы измерения	Значения	
			Аналог	Проект
Расходы на содержание обслуживающего персонала	Z	руб/год	96480	48240
Капитальные затраты на этапе проектирования	K_p	руб/год	-	104046,608

Продолжение таблицы 4.6

Экономические показатели	Буквенное обозначение	Единицы измерения	Значения	
			Аналог	Проект
Расходы на функционирование программы	М	руб/год	24192	12096
Эксплуатационные расходы	р	руб/год		121770,72
Экономия за счет повышения производительности труда	ΔР	руб/год		48240
Годовая экономия	Эд	руб		170010,72
Ожидаемый экономический эффект	Э	руб/год		154403,7288
Цена программного продукта	Ц	руб		119653,5992
Срок окупаемости	т	года		0,61

5 Охрана труда и окружающей среды

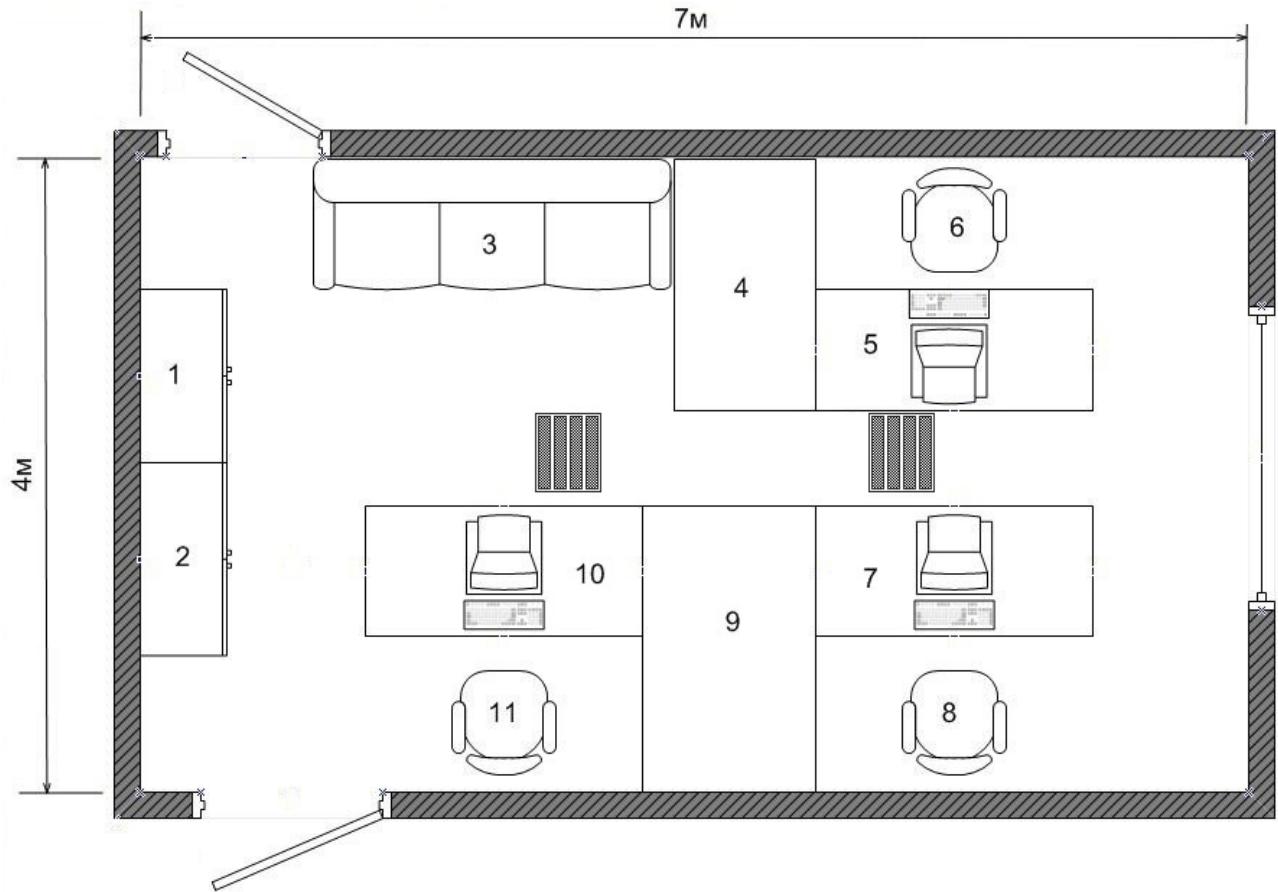
Данный раздел состоит из двух частей. Первая часть — аттестации рабочего места программиста. Вторая часть — рассмотрение альтернативных решений разрабатываемого продукта с точки зрения влияния на производительность труда.

5.1 Аттестация рабочего места программиста

При аттестации рабочих мест проводится, по существу, аудит условий труда на рабочих местах предприятия сторонней аккредитованной организацией. Результаты такого независимого аудита могут быть эффективно использованы работодателем для документально обоснованных его действий по всем направлениям обеспечения безопасных условий и охраны труда согласно Трудовому Кодексу РФ.

При вступлении России в ВТО товары отечественных предприятий смогут реально конкурировать с импортными товарами на внутреннем рынке (не говоря уже о внешнем) только случае, если на этих отечественных предприятиях будет сертифицирована по международным стандартам система качества. Составная часть сертификации системы качества - сертификация работ по охране труда на предприятиях, начальным этапом которой является аттестация рабочих мест.

Рассмотрим рабочее помещение программиста. АлтГТУ, проспект Ленина, 46 ауд.303а.



1,2 — шкаф, 3 — диван, 4,9 — стол, 5,7,10 — рабочий стол, 6,8,11 — стулья.

Рисунок 5.1 - Эскиз помещения

Рассмотрим факторы повышенного риска, к которым отнесем шум, освещение, неионизирующие поля и излучения, напряженность труда.

5.1.1 Шум

Существующие на рабочем месте источники шума:

Внутренний шум (коридор, учебные аудитории) $L_{\text{Э1}} = 30 \text{ дБА}$

Системный блок (кулер) $L_{\text{Э2}} = 32 \text{ дБА}$

Помещение для крупного телекоммуникационного или серверного оборудования. $L_{\text{Э3}} = 38 \text{ дБА}$

Время воздействие перечисленных источников: постоянно.

Определим эквивалентный уровень шума $L_{\text{Э2,1}}$ по формуле (5.1).

$$L_{\text{Э3,2}} = L_{\text{Э3}} + \Delta L_1 = 38 + 1 = 39 \text{ дБА}, \quad (5.1)$$

где $L_{\text{Э3,2}}$ — эквивалентный уровень шума;

$L_{\text{Э3}}$ — величина шума;

По таблице П.11.1 руководства Р 2.2.2006-05 определяется ΔL_1 по разнице шумов $L_{\text{Э}2}$ и $L_{\text{Э}3}$ в 6 дБА. $\Delta L_1 = 1$ дБ.

Суммарный уровень шума можно рассчитать по следующей формуле

$$L_{\text{Э}3,2,1} = L_{\text{Э}3,2} + \Delta L_2 = 39 + 0,5 = 39,5 \approx 40 \text{ дБА}, \quad (5.2)$$

где $L_{\text{Э}3,2,1}$ – суммарный уровень шума;

$L_{\text{Э}3,2}$ – величина эквивалентного шума (формула (5.1));

$L_{\text{Э}1}$ – величина шума;

По таблице П.11.1 руководства Р 2.2.2006-05 определяется ΔL_2 по разнице шумов $L_{\text{Э}3,2}$ и $L_{\text{Э}1}$ в 9 дБА. $\Delta L_2 = 0,5$ дБ.

Сравнивая значения таблицы 1 в СН 2.2.4_2.1.8.562-96 и результаты расчета по формуле (5.2), в соответствии с руководством Р2.2.2006-05 (таблица 4) уровень шума не превышает ПДУ, что соответствует **2 классу (допустимому)**.

5.1.2 Искусственная освещенность

По СНиП 23-05-95* (таблица 2) при наименьшем размере объекта различия от 0,3 до 0,5 мм определяем разряд зрительных работ: Б. Так как относительная продолжительность зрительной работы при направлении зрения на рабочую поверхность составляет 70%, мы относим зрительные работы к подразряду «1», что соответствует нормированной освещенности при искусственном освещении $E_h=300$ лк.

Освещение рабочего места обеспечивается 2 светильниками, состоящими из 4 лампы Philips TL-D 18W/33 SLV (газоразрядная ртутная лампа низкого давления с трубчатой колбой диаметром 26 мм). Снизу лампы не закрыты. Световой поток одного одной лампы $F_{\text{лм}}= 1200$ лм. Световой поток одного светильника $F_{\text{св}}= 4 * F_{\text{лм}}= 4 * 1200 = 4800$ лм.

Освещенность рабочего места рассчитывается по следующей формуле

$$E = \frac{F * n * n}{S * k * z} = \frac{4800 * 2 * 0,69}{28 * 1,4 * 1,0} \approx 169 \text{ лк}, \quad (5.3)$$

где E – значение освещенности;

F – световой поток светильника, лм.;

n – количество светильников;

A = 7 м. - длина помещения;

B = 4 м. - ширина помещения;

S = A*B=28 м² - площадь помещения;

H_n =2,5 м. - высота подвеса люстры над рабочей плоскостью;

k = 1,4 – коэффициент запаса (определен по таблице 7 пособия Малкиной Е.М. “Расчет искусственной освещенности в производственных помещениях” для помещений общественных и жилых зданий);

z – коэффициент неравномерности освещения, z = 1,0 (определен по таблице 6 пособия Малкиной).

η – коэффициент использования светильником установки;

Функция коэффициента использования светильником установки приведена в следующей формуле

$$n = f(T, i, p_n, p_{cm}, p_{pn}), \quad (5.4)$$

где Т – тип светильника;

i – индекс помещения;

p_n= 70% – коэффициент отражения света от потолка;

p_{cm}= 60% – коэффициент отражения света от стен;

p_{pn}= 60% – коэффициент отражения света от рабочей поверхности.

Соотношение размеров освещаемого помещения определяется индексом помещения i по формуле из пособия Малкиной Е.М. “Расчет искусственной освещенности в производственных помещениях”.

$$i = \frac{A*B}{H_n + (A+B)} = \frac{4*7}{2,5*(4+7)} = 1,018 \approx 1, \quad (5.5)$$

При таком индексе помещения i = 1 и коэффициентах отражения, коэффициент использования светового потока определен методом интерполяции и равен η = 69 % (Приложение 2, пособия Малкиной).

Результат вычислений по формуле (5.5) составляет больше половины E_н и меньше E_н, и в соответствии с руководством Р2.2.2006-05 (таблица 12) условия труда по параметрам освещенности соответствуют **3 классу (вредному) 1 степени.**

5.1.3 Неионизирующие электромагнитные поля и излучения

Самым мощным источником в компьютере является ЭЛТ монитор (монитор с электроннолучевой трубкой), но их вытеснили ЖК-мониторы, сила излучения, которых не выше фоновой. Таким образом в целом излучение компьютера не выше естественного фона излучения.

Согласно руководству Р 2.2. 2006 - 05 (таблица 15) условия труда соответствуют **2 классу (допустимому)**.

В настоящее время о влиянии электромагнитного излучения на организм человека, практически ни чего не известно, да и за компьютерами мы сидим пока лет 20. Однако некоторые работы и исследования в этой области определяют возможные факторы риска, так например, считается что электромагнитное излучение может вызвать расстройства нервной системы, снижение иммунитета, расстройства сердечнососудистой системы и аномалии в процессе беременности и соответственно пагубное воздействие на плод.

5.1.4 Напряженность трудового процесса

Используя таблицу 18 руководства Р 2.2.2006 — 05, заполним таблицу 5.1.

Таблица 5.1 – Классы условий труда по показателям напряженности трудового процесса

Показатели	Класс условий труда				
	1	2	3.1	3.2	3.3
Содержание работы					
Восприятие сигналов и их оценка	+				
Распределение функции по степени сложности задания		+			
Характер выполняемой работы		+			
Длительность сосредоточенного наблюдения		+			
Плотность сигналов за 1 час работы	+				
Число объектов одновременного наблюдения		+			

Продолжение таблицы 5.1

Показатели	Класс условий труда				
	1	2	3.1	3.2	3.3
Размер объекта различения при длительности сосредоточенного внимания				+	
Работа с оптическими приборами при длительности сосредоточенного наблюдения	+				
Наблюдение за экраном видеотерминал			+		
Нагрузка на слуховой анализатор	+				
Нагрузка на голосовой аппарат	+				
Степень ответственности за результат собственной деятельности.				+	
Значимость ошибки.					
Степень риска для собственной жизни	+				
Ответственность за безопасность других лиц	+				
Количество конфликтных производственных ситуаций за смену	+				
Число элементов (приемов), необходимых для реализации простого задания или в многократно повторяющихся операциях		+			
Продолжительность выполнения простых заданий или повторяющихся операций	+				
Время активных действий			+		
Монотонность производственной обстановки	+				
Фактическая продолжительность рабочего дня		+			
Сменность работы	+				
Наличие регламентированных перерывов и их продолжительность	+				
Количество показателей в каждом классе	13	6	3	1	0

Из таблицы 5.1 видно, что наибольший класс условий труда соответствует третьему степени 2, но он встречается не более 2-х раз, в соответствии с руководством Р 2.2.2006 - 05 общая оценка условий труда по напряженности трудового процесса соответствует **3 классу (вредному) 2 степени.**

5.1.5 Взрывопожаробезопасность

В помещении находится следующая мебель:

2 шкафа из древесностружечной плиты (ДСП)

5 столов из древесностружечной плиты (ДСП)

3 стула в изготовлении которого использовались метал, пластмасса и немного ткани

1 диван — металлический каркас, обшитый кожным заменителем и поролоновым наполнителем.

Для определения пожароопасной категории помещения необходимо подсчитать пожарную нагрузку Q по формуле

$$Q = \sum G_i Q_{hi}^p, \quad (5.6)$$

где G_i — количество i -го материала пожарной нагрузки, кг;

Q_{hi}^p — низшая теплота сгорания i -го материала пожарной нагрузки, Мдж*кг-1.

Также необходимо подсчитать удельную пожарную нагрузку g по формуле

$$g = Q/S \quad (5.7)$$

где S — площадь размещения пожарной нагрузки, м² (но не менее 10 м²).

Для определения низшей теплоты сгорания в таблице 5.2 приведены ее значения для некоторых материалов.

Таблица 5.2 — Низшая теплота сгорания веществ

Вещества и материалы	Низшая теплота сгорания Q_{hi}^p , Мдж/кг
Бензин	41,87
Бумага: книги, журналы	13,40
Древесина (брюски $W = 14\%$)	13,80
Линолеум: поливинилхлоридный	14,31
Резина	33,52
Полиэтилен	47,14
Древесина в изделиях	16,6

Суммарная масса древесных изделий составляет 250 кг, линолеума — 30 кг, бумаги (обои, документы) — 10 кг.

Подсчитаем пожарную нагрузку по формуле (1)

$$Q = 250 * 16,6 + 30 * 14,31 + 10 * 13,4 = 4713,3 \text{ МДж}$$

Теперь подсчитаем удельную пожарную нагрузку по формуле (5.7)

$$g = 4713,3 / 28 = 168 \text{ МДж} \cdot \text{м}^{-2}$$

В соответствии с НПБ 105-03 (таблица 4) помещения, в которых удельная пожарная нагрузка составляет не больше $180 \text{ МДж} \cdot \text{м}^{-2}$ равна В4.

Так как площадь этажа между противопожарными стенами 1-го типа, определяемыми по ГОСТ Р 12.3.047-98 таблица У.1, меньше 2000 м^2 и в соответствии с СНиП 2.08.02-85 степень огнестойкости равна III.

Пожарная профилактика представляет собой комплекс организационных и технических мероприятий, направленных на обеспечение безопасности людей, на предотвращении пожара, ограничение его распространения, а также создание условий для успешного тушения пожара. Для профилактики пожара чрезвычайно важна правильная оценка пожароопасности здания, определение опасных факторов и обоснование способов и средств пожаропредупреждения и защиты.

Одно из условий обеспечения пожаробезопасности - ликвидация возможных источников воспламенения.

В лаборатории источниками воспламенения могут быть:

- неисправное электрооборудование, неисправности в электропроводке, электрических розетках и выключателях. Для исключения возникновения пожара по этим причинам необходимо вовремя выявлять и устранять неисправности, проводить плановый осмотр и своевременно устранять все неисправности;
- неисправные электроприборы. Необходимые меры для исключения пожара включают в себя своевременный ремонт электроприборов, качественное исправление поломок, не использование неисправных электроприборов;

- обогревание помещения электронагревательными приборами с открытыми нагревательными элементами. Открытые нагревательные поверхности могут привести к пожару, так как в помещении находятся бумажные документы и справочная литература в виде книг, пособий, а бумага – легковоспламеняющийся предмет. В целях профилактики пожара предлагаю не использовать открытые обогревательные приборы в помещении лаборатории;
- короткое замыкание в электропроводке. В целях уменьшения вероятности возникновения пожара вследствие короткого замыкания необходимо, чтобы электропроводка была скрытой.
- попадание в здание молнии. В летний период во время грозы возможно попадание молнии вследствие чего возможен пожар. Во избежание этого рекомендуется установить на крыше здания молниеприемник;
- несоблюдение мер пожарной безопасности и курение в помещении также может привести к пожару.

В целях предотвращения пожара предлагаю проводить с инженерами противопожарный инструктаж, на котором ознакомить работников с правилами противопожарной безопасности, а также обучить использованию первичных средств пожаротушения.

В случае возникновения пожара необходимо отключить электропитание, вызвать по телефону пожарную команду, эвакуировать людей из помещения согласно плану эвакуации, и приступить к ликвидации пожара огнетушителями. При наличии небольшого очага пламени можно воспользоваться подручными средствами с целью прекращения доступа воздуха к объекту возгорания.



Рисунок 5.2 - План эвакуации

5.1.6 Электробезопасность

В помещении 4 электрические розетки с заземлением с наклейками номинального напряжения подоваемого на них. На рабочем месте программиста из всего оборудования металлическим является лишь корпус системного блока компьютера, но здесь используются системные блоки, отвечающие стандарту фирмы IBM, в которых кроме рабочей изоляции предусмотрен элемент для заземления и провод с заземляющей жилой для присоединения к источнику питания.

К основным причинам поражения программиста электрическим током на рабочем месте можно отнести:

Прикосновение к металлическим нетоковедущим частям системного блока ПЭВМ, которые могут оказаться под напряжением в результате повреждения изоляции.

Запрещенное использование электрических приборов, таких как электрические плиты, чайники, обогреватели.

Все токоведущие части ЭВМ изолированы, то случайное прикосновение к токоведущим частям исключено, а запрещенные электрические приборы не используются.

Влажность в помещении не превышает 60% и технологическая пыль отсутствует, поэтому по ПУЭ помещение относится к сухому и не пыльному. Полы в помещении не являются токопроводящими, температура контролируется автоматической системой кондиционирования и колеблется в пределах от 22 до 24 градусов Цельсия. Системные блоки компьютеров расположены на специальных полках компьютерных столов, что препятствует возможности одновременного прикосновения человека к металлоконструкциям зданий и к металлическим корпусам электрооборудования.

Из всего вышесказанного и согласно ПЭУ от 08.07.2002 рассматриваемое помещение можно отнести к 1 классу помещений без повышенной опасности.

5.1.7 Травмобезопасность

В помещении находится следующее оборудование:

- 3 компьютера
- 3 компьютерных стола
- 2 рабочих стола
- 3 стула
- 2 шкафа
- диван

Травмобезопасность рабочих мест обеспечивается исключением повреждений частей тела человека, которые могут быть получены в результате воздействия:

- движущихся предметов, механизмов или машин, а также неподвижными их элементами на рабочем месте (при механическом воздействии). Такими предметами являются: зубчатые, цепные, клиноременные передачи, кривошипные

механизмы, подвижные столы, вращающиеся детали, органы управления и т.п.;

- электрического тока. Источником поражения могут быть незащищенные и неизолированные электропровода, поврежденные электродвигатели, открытые коммутаторы, незаземленное оборудование и др.;
- агрессивных и ядовитых химических веществ. Например, химические ожоги сильными кислотами, едкими щелочами и ядовитыми химическими веществами (хлор, аммиак, и т.д.) при попадании их на кожу или в легкие при вдыхании;
- нагретых элементов оборудования, перерабатываемого сырья, других теплоносителей (при термическом воздействии). Примерами таких элементов являются горячие трубопроводы, крышки котлов, танков, корпуса оборудования, детали холодильных установок и т.д.;
- а также повреждения, полученные при падениях. Падения подразделяются на два вида: падения на человека различных предметов и падения человека в результате подскользывания, запинания, падения с высоты или внезапного ухудшения здоровья.

Из движущихся предметов на рабочем месте имеется только стул, но его масса слишком мала чтобы причинить вред здоровью. Электробезопасность уже была рассмотрена выше. Из ядовитых веществ на рабочем месте имеется только тонер в лазерном принтере, но принтер печатает не много и помещение часто проветривается, поэтому влияние минимально. На рабочем месте не используются нагревательные элементы. Вероятность получить травму в результате падения высокая, так как мебели на рабочем месте много и расположена она не удачно и много острых углов.

Все оборудование находится в исправном техническом состоянии, розетки для подключения электрооборудования имеют заземление и подписаны

соответственно номинальному напряжению. Также в помещении находится инструкция по технике безопасности.

По МУ ОТ РМ 02-99 условия труда по травмобезопасности относятся к 1 классу (оптимальные).

5.2 Обзор альтернативных программных решений с точки зрения повышения производительности труда

5.2.1 Обзор системы Zabbix

В любой сети, где есть больше, чем один сервер, очень полезно бывает иметь перед глазами полную картину происходящего. В крупных сетях, где количество хостов переваливает за несколько десятков, следить за каждым в отдельности — непосильная задача для администраторов. Для облегчения задачи наблюдения применяются системы мониторинга и одной из них является Zabbix.

5.2.1.1 Общие сведения

Zabbix создан Алексеем Владышевым и в настоящее время активно разрабатывается и поддерживается ZabbixSIA.

Zabbix это открытое решение распределенного мониторинга корпоративного класса.

Zabbix это программное обеспечение мониторинга многочисленных параметров сети а также состояния и работоспособности серверов. Zabbix использует гибкий механизм уведомлений, что позволяет пользователям настраивать оповещения по почте практически для любого события. Это дает возможность быстро среагировать на проблемы с сервером. Zabbix предлагает отличные возможности отчетности и визуализации данных, базируясь на собранных данных. Это делает Zabbix идеальным инструментом для планирования и масштабирования.

Zabbix написан и распространяется под лицензией GPLv2. Это означает, что его исходный код свободно распространяется и доступен широкой публике.

Так же доступна коммерческая поддержка, которая предоставляется компанией Zabbix.

Zabbix предлагает:

- автоматическое обнаружение серверов и других устройств в сети
- распределенный мониторинг с централизованным администрированием через ВЕБ
- поддержка обоих механизмов пуллеров и трапперов
- серверное программное обеспечение для Linux, Solaris, HP-UX, AIX, FreeBSD, OpenBSD, OSX
- родные агенты с высокой производительностью (клиентское программное обеспечение для Linux, Solaris, HP-UX, AIX, FreeBSD, OpenBSD, OSX, Tru64/OSF1, WindowsNT4.0, Windows 2000, Windows 2003, WindowsXP, WindowsVista)
- мониторинг без агентов
- безопасная аутентификация пользователей
- гибкая система прав доступа пользователей
- Web-интерфейс
- гибкая система уведомлений по e-mail о предопределенных событиях
- высокоуровневый (класса “Бизнес”) вид контроля ресурсов
- Открытое программное обеспечение
- агенты с высокой эффективностью для UNIX и WIN32 платформ
- легкость в изучении
- увеличивает рентабельность (простои очень дорого обходятся)
- низкая стоимость обслуживания
- очень простое конфигурирование
- централизованная система мониторинга. Вся информация (конфигурация и данные о производительности) хранится в реляционной базе данных

- высокоуровневое дерево предоставляемых услуг
- очень простая установка
- поддержка SNMP (v1,v2,v3). Оба режима пуллера и траппера.
- возможность визуализации
- встроенный механизм очистки устаревших данных

5.2.1.2 Описание основных функций

Система состоит из нескольких частей, и при большой нагрузке и наблюдении за очень большим количеством хостов позволяет разнести эти части на несколько раздельных машин.

Zabbix состоит из

- собственно сервера мониторинга, который выполняет периодическое получение данных, обработку, анализ и запуск скриптов оповещения
- базы данных (MySQL, PostgreSQL, SQLite или Oracle)
- веб-интерфейса на PHP
- агента — демона, который запускается на отслеживаемых объектах и предоставляет данные серверу. Агент опционален, мониторинг можно производить не только с помощью него, но и по SNMP (версий 1, 2, 3), запуском внешних скриптов, выдающих данные, и несколько видов предопределенных встроенных проверок, таких как ping,

Основная логическая единица — Узлы сети (host), сервера, находящиеся под наблюдением. Каждому серверу присваивается описание и адрес (dns или ip, можно оба, причем возможностью выбирать, что использовать для соединения).

Узлы объединяются в группы, например веб-сервера или сервера баз данных. Группы служат для вывода только определенных серверов при наблюдении.

Каждый узел имеет несколько Элементов данных (items) — параметров, за которыми ведется мониторинг. К примеру, на всех серверах у меня есть параметр ping, (он получается с помощью встроенной проверки), который равняется 1, если

ответ на последний ping-запрос был получен, иначе 0. А на одном из серверов у меня есть параметр «количество пользователей онлайн», который собирается самописным скриптом из базы данных сайта. Для каждого элемента данных можно указать свой период обновления, способ хранения (сам параметр или скорость его изменения), множитель, временной интервал сбора (например, только в рабочее время).

Создавать элементы данных для каждого из множества серверов — сложно, поэтому можно создать узлы-шаблоны. Эти узлы тоже содержат элементы данных, но они не мониторятся напрямую. Вместо этого реальный хост связывается с одним или несколькими шаблонами, и все параметры шаблона автоматически наследуются хостом.

Zabbix предоставляет гибкие возможности по настройке условий-триггеров, которые включаются при авариях и неполадках, и система начинает сигнализировать администратору, что что-то случилось. При включении триггера веб-интерфейс начинает издавать звуковые сигналы.

Time	Description	Status	Severity	Duration	Ack	Actions
2008.Sep.18 14:59:12	Processor load is too high on au_001	OK	Warning	3h 8m 49s	No	Failed
2008.Sep.18 14:58:41	Processor load is too high on au_001	PROBLEM	Warning	81s	No	Failed
2008.Sep.18 14:52:39	Processor load is too high on au_001	OK	Warning	6m 2s	No	Failed
2008.Sep.18 14:52:10	Processor load is too high on au_001	PROBLEM	Warning	29s	No	Failed
2008.Sep.18 14:48:40	Processor load is too high on au_001	OK	Warning	3m 30s	No	Failed
2008.Sep.18 14:48:15	Processor load is too high on au_001	PROBLEM	Warning	25s	No	Failed
2008.Sep.18 14:47:38	Processor load is too high on au_001	OK	Warning	37s	No	Failed
2008.Sep.18 14:46:12	Processor load is too high on au_001	PROBLEM	Warning	1m 26s	No	Failed
2008.Sep.18 14:41:11	Processor load is too high on au_001	OK	Warning	5m 1s	No	Failed
2008.Sep.18 14:40:40	Processor load is too high on au_001	PROBLEM	Warning	31s	No	Failed
2008.Sep.18 14:32:13	Processor load is too high on au_001	OK	Warning	8m 27s	No	Failed
2008.Sep.18 14:31:43	Processor load is too high on au_001	PROBLEM	Warning	30s	No	Failed
2008.Sep.18 14:24:12	Processor load is too high on au_001	OK	Warning	7m 31s	No	Failed
2008.Sep.18 14:23:41	Processor load is too high on au_001	PROBLEM	Warning	31s	No	Failed
2008.Sep.18 14:22:10	Processor load is too high on au_001	OK	Warning	1m 31s	No	Failed
2008.Sep.18 14:21:12	Processor load is too high on au_001	PROBLEM	Warning	58s	No	Failed
2008.Sep.18 14:15:43	Processor load is too high on au_001	OK	Warning	7m 29s	No	Failed
2008.Sep.18 14:13:12	Processor load is too high on au_001	PROBLEM	Warning	31s	No	Failed
2008.Sep.18 14:00:09	Processor load is too high on au_001	OK	Warning	13m 3s	No	Failed
2008.Sep.18 13:59:14	Processor load is too high on au_001	PROBLEM	Warning	55s	No	Failed

Рисунок 5.3 – Результат выполнения ping

У Zabbix есть возможность отправить уведомление на почту, в jabber или sms с помощью gsm-модема, или даже попытаться самостоятельно восстановить сервис

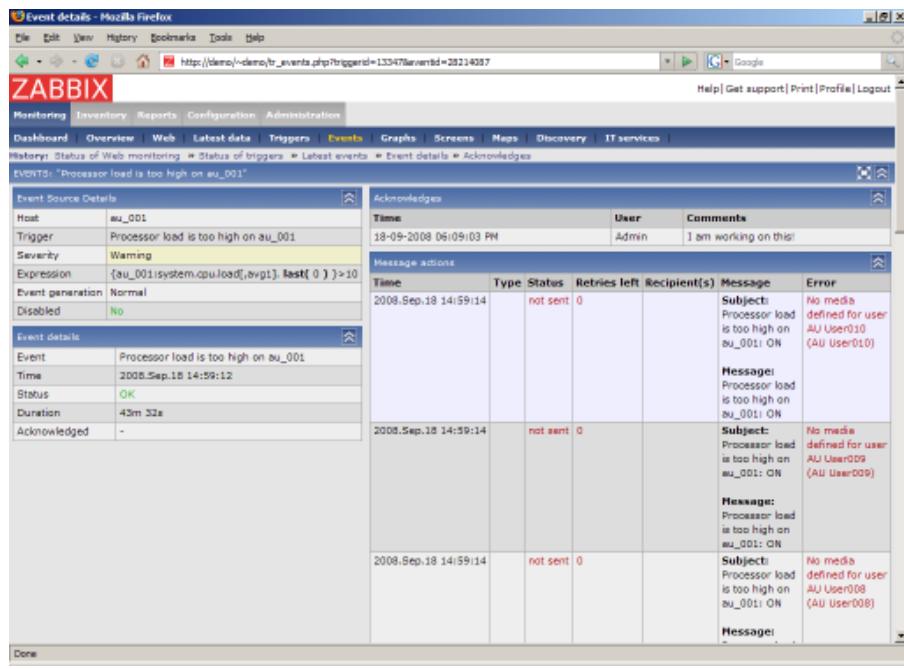


Рисунок 5.4 – Интерфейс настройки триггеров

По данным любого параметра система сможет построить график изменения за любой промежуток времени с максимальным разрешением. Имеется возможность создавать сложные графики, отображающие на одном поле несколько параметров

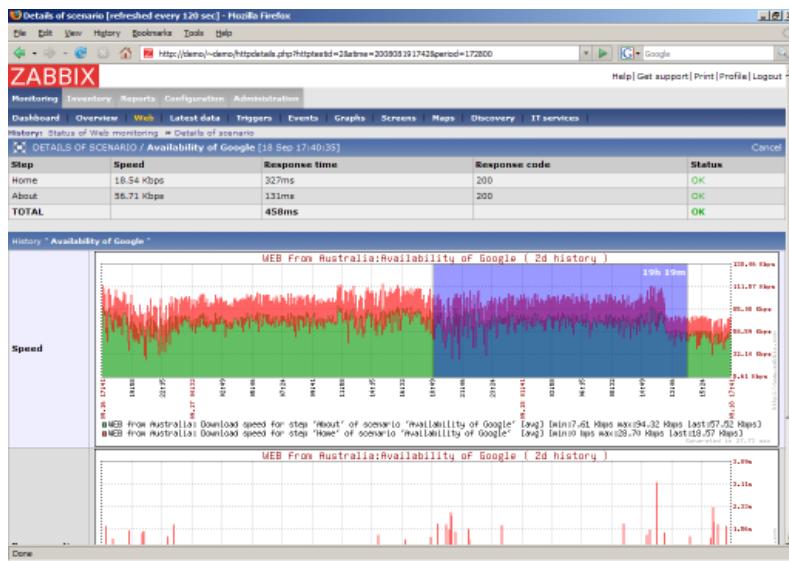


Рисунок 5.5 – График построенный по результатам LoadAverage

Для отображения логической структуры сети имеется возможность создавать карты сети, отображающие именно расположение узлов сети и связей между ними. Естественно, состояние узлов (доступен или нет) отображается и на карте.

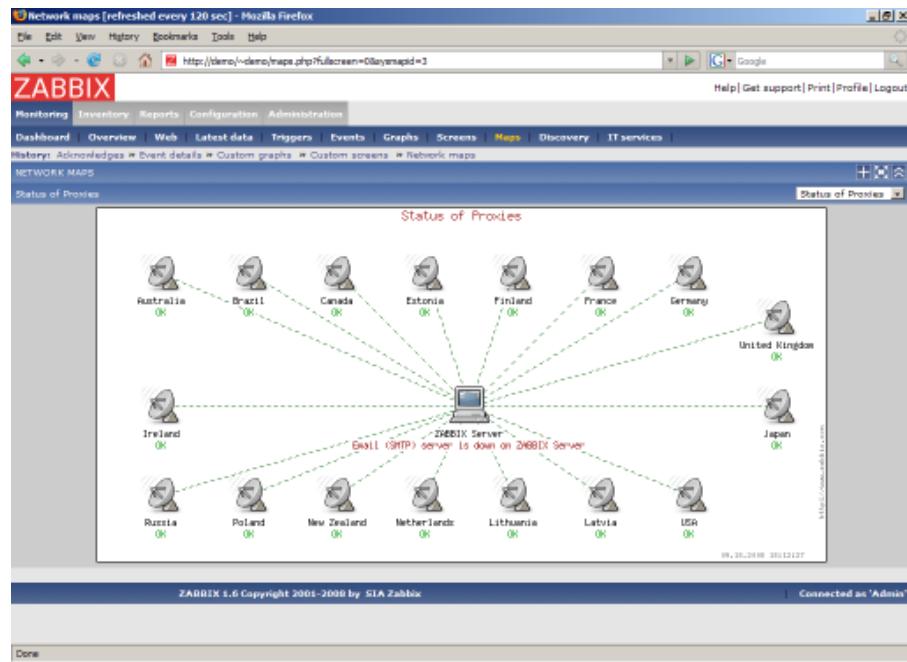


Рисунок 5.6 - Карта состояния узлов

Имеются комплексные отчеты, которые позволяют на одном экране просматривать сразу несколько сущностей — графики, данные, триггеры...

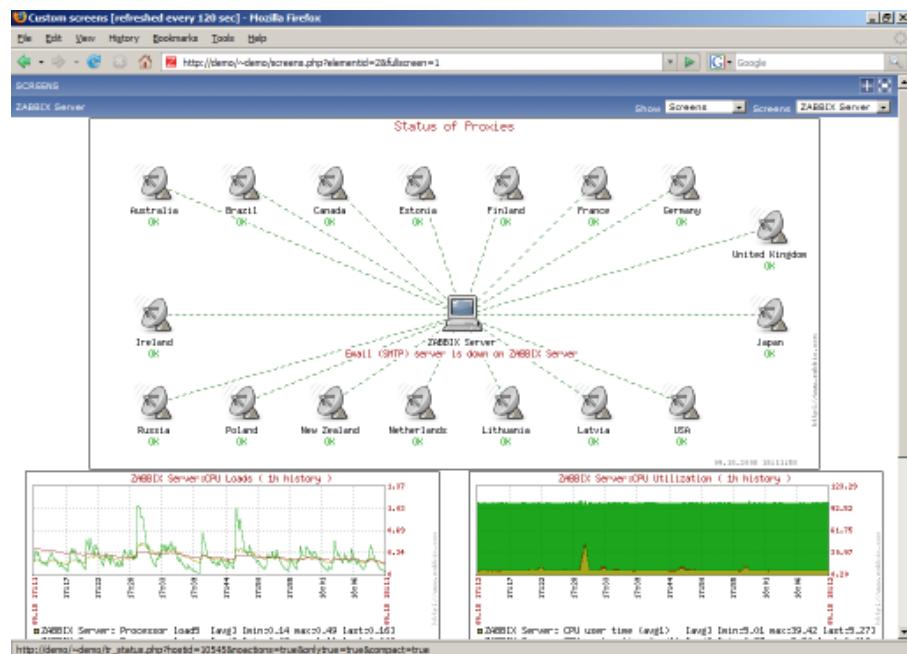


Рисунок 5.7 – Комплексный отчет

Zabbix — довольно мощная и обширная система, и запас у него есть еще полдесятка функций, которые позволяют еще больше упростить наблюдение за сетью, такие как мониторинг состояния веб-сайта с помощью автоматического выполнения сценария.

5.2.1.3 Удобство использования

Веб-интерфейс консоли управления реализован через обращение PHP скриптов напрямую к СУБД. Поддерживается SSL. Автоматическое отсоединение после 30 минут бездействия. После 5 неудачных попыток доступ блокируется на 1 минуту, а IP-адрес показывается настоящему администратору.

Консоль отображает состояние контролируемых объектов и параметров, состояние триггеров, историю событий и реакций оператора, карту сети и графики изменения параметров.

Можно выбрать несколько объектов из списка, нажимая Ctrl одновременно с указанием мышью.

Каждый пользователь может подогнать вид консоли под свои нужды - поменяв главную панель, создав свои графики, карты, экраны и прочее.

Кроме встроенных графиков по каждому числовому параметру можно определять свои составные графики: имя, ширина, высота, тип (нормальный, стек, пирог, 3D пирог), показывать ли границы рабочего времени (настраивается в общих настройках), показывать ли границы триггеров, тип оси Y (автоматически, положительное автоматически, вручную), элементы данных (имя, предобработка, что делать при наличии нескольких данных в одном пикселе, цвет, тип линии или заполнения, приоритет сортировки - с меньшим числом будет внизу стека). Нельзя составлять выражения из элементов данных.

Карта позволяет наглядно показать связь хостов между собой, картинку (общие размеры задаются в пикселях) необходимо рисовать самостоятельно, задавая вручную координаты (в пикселях) иконок хостов на плоскости и какой хост с каким соединён. Новые иконки (встроенные ужасны) загружаются в общих настройках (PNG, 48x48 или 24x24). Там же можно загружать фоновые картинки

(масштабирование при визуализации не производится). Рекомендуется предварительно задуматься о прозрачности. Удаления иконок нет. В общих настройках карты указывается тип подписей (указанную при составлении карты подпись и статус, IP адрес и статус, имя хоста и статус, только статус хоста или совсем ничего) к иконкам и где их размещать по умолчанию.

Для каждого элемента карты указываются

- тип элемента карты
- иконка изображает состояние всех триггеров хоста (Host, узел сети)
- иконка изображает состояние всех элементов карты
- иконка изображает состояние триггера
- иконка изображает состояние всех триггеров всех хостов группы
- ни с чем не связанная иконка
- подпись к иконке (общие настройки карты)
- расположение подписи, если не по умолчанию
- хост, группа или триггер в зависимости от типа элемента карты
- имя иконки для объекта в нормальном состоянии, при наличии проблемы, при неопределённом состоянии, при отключённом объекте
- URL, который будет связан с иконкой (иначе будет доступ к скриптам и триггерам хоста)

Для каждого соединения элементов карты указываются

- элементы карты, которое оно соединяет
- тип и цвет линии для нормального состояния
- набор триггеров, позволяющих изменить тип линии и цвет соединения

К узлам карты могут быть приписаны скрипты (Администрирование - Скрипты). При задании команды можно использовать макросы. В комплекте идут ping и traceroute (запускаются на сервере, результат в окне браузера). В

настройках скриптов можно задать допустимую группу пользователей, группу хостов и наличие прав на чтение или запись.

Экран (Screen, комплексный отчёт) позволяет собрать свою страницу из карт, графиков, триггеров и прочего. Экран представляет собой прямоугольную таблицу (можно сливать ячейки по горизонтали и вертикали), в ячейках которой можно размещать:

- часы
- обзор данных о группах хостов
- встроенные графики
- определённые пользователем графики
- информацию о хостах
- карты
- строки текста
- обзор состояния сервера
- общий обзор триггеров
- обзор триггеров для группы хостов
- историю событий
- историю действий
- вложенные экраны
- вложенные произвольные страницы (URL)

Ячейки можно сливать по горизонтали и вертикали, выравнивать содержимое по горизонтали и вертикали.

Из экранов можно делать слайд-шоу, задав последовательность экранов и интервал демонстрации каждого из них.

Инвентаризация - поиск и просмотр профилей узлов (модель, серийный номер и пр.). Увы - их придётся в начале задать.

Есть аудит действий системы (action) и действий администратора (вход, выход, добавление, удаление, изменение).

5.2.1.4 Повышение производительности управление учетными записями

Имеется управление пользователями и правами доступа. Методы аутентификации: собственная (имена и пароли, хранятся в БД в зашифрованном виде), от Apache, LDAP. В случае аутентификации Apache и LDAP пользователь в смысле Zabbix тоже должен существовать, но его пароль не используется. При использовании LDAP пользователи из группы, для которой указан метод доступа "Internal", будут аутентифицироваться локально. Типы пользователей: ZABBIXUser (доступ к меню мониторинга, по умолчанию не имеет доступа ни к каким хостам и группам, необходимо предоставлять доступ явно), ZABBIXAdmin (доступ к меню мониторинга и конфигурации, по умолчанию не имеет доступа ни к каким хостам и группам, необходимо предоставлять доступ явно), ZABBIXSuperAdmin (доступ к меню мониторинга, конфигурации и администрирования; имеет неотзывный доступ на чтение и запись ко всем хостам и группам). Для пользователя также задаются: входное имя, имя собственное, фамилия, список групп, среда передачи сообщений (Media), язык (русский есть), тема оформления, использование куки для автоматического входа, выход по неактивности, начальный URL, интервал обновления экрана. Здесь же можно посмотреть права доступа, определяемые членством в группах. Пользователь может самостоятельно настроить (в профиле): пароль, язык, тему оформления, использование куки для автоматического входа, начальный URL, интервал обновления экрана. По умолчанию создаются пользователи Admin (максимальные права) и Guest (минимальные права, в этом режиме с системой могут работать незарегистрированные пользователи). Управление правами доступа осуществляется с помощью включения пользователя в группы пользователей и задании хостов, к которым пользователи группы могут иметь доступ на чтение и запись. Для членов группы определяется: список пользователей, доступность веб-интерфейса (включить, выключить, системные умолчания), заблокированность пользователя, права на запись/чтение и чтение к группам хостов, группы, доступ к

которым запрещён. Можно создавать свои группы пользователей, и имеются группы пользователей по-умолчанию:

- Zabbix administrators (члены этой группы получают извещения о проблемах с СУБД)
- UNIX administrators
- WEB administrators
- Security specialists
- Network administrators
- Head of IT department
- Disabled (сюда надо помещать временно заблокированных пользователей вместо их удаления)
- Database administrators

5.2.1.5 Поддерживаемые платформы

Таблица 5.3 – Поддерживаемые платформы Zabbix

Платформа	Zabbix-сервер	Zabbix-агент
AIX	Поддерживается	Поддерживается
FreeBSD	Поддерживается	Поддерживается
HP-UX	Поддерживается	Поддерживается
Linux	Поддерживается	Поддерживается
Mac OS X	Поддерживается	Поддерживается
Novell Netware	-	Поддерживается
OpenBSD	Поддерживается	Поддерживается
SCO OpenServer	Поддерживается	Поддерживается
Solaris	Поддерживается	Поддерживается
Tru64/OSF	Поддерживается	Поддерживается
Windows NT 4.0, Windows 2000, Windows 2003, Windows XP, Windows Vista	-	Поддерживается

5.2.2 Превосходство над аналогами

В качестве альтернативных программных решений будем рассматривать такие программные продукты как Zabbix, рассмотренный выше, Nagios, Cacti, т. к. эти решения являются прямыми конкурентами разрабатываемого продукта.

Разрабатываемый продукт имеет два основных весомых достоинства перед аналогичными программами:

- распределенность
- дешевизна расширения

Распределенность системы заключается в том, что все узлы равноправны и их количество не ограничено каким-то значением. Среди узлов нет «главных», а значит нет центра из которого все регулируется, а значит и нет опасности потерять с ним связь и терять драгоценное процессорное время.

А раз узлы равноправны то и подключение и отключение узлов можно осуществлять в любом порядке и в любом количестве.

Дешевизна расширения заключается в том, что подключение нового узла ничтожно по себестоимости. Если подключаемый узел активный(т.е. обладает своими вычислительными мощностями), то ничего дополнительного не нужно. Просто устанавливается часть ядра системы и узел готов к работе.

Помимо двух основных преимуществ есть еще:

- a) Немаловажным преимуществом является полная кросплатформенность.
- b) Возможность управлять системой из любого узла системы
- c) Абсолютная свобода в выборе и написании задач для выполнения на узлах.

5.3 Выводы

После всего выше сказанного можно сделать выводы

Рассматриваемые альтернативные решения уже какое-то время присутствуют на рынке и уже заняли свою нишу. Имеют свою плюсы и минусы, повышает эффективность труда работника.

Разрабатываемый продукт лучше как минимум в двух рассматриваемых категориях.

Рассмотренные превосходства над аналогичными решениями не только повышают производительность труда, но и снижают финансовые затраты.

Разрабатываемый продукт очень гибкий в использовании инструмент, поэтому он может быть использован большим числом работников для своих нужд.

Заключение

В результате работы над дипломным проектом была выдвинута и исследована идея построения распределенной системы мониторинга гетерогенной среды. Были изучены аналоги в сфере современных систем мониторинга, произведена оценка их эффективности и применимости согласно выдвинутой модели требований. В следствии чего, были сделаны выводы о необходимости появления нового класса инструментов мониторинга, в виду неготовности существующих решений удовлетворять видвигаемым к ним требованиям.

Кроме того, была разработана и формализована модель распределенной системы мониторинга, лишенная недостатков классических клиент-серверных систем и полностью удовлетворяющая видвинутой модели требований.

На основании формализованной модели был спроектирован и реализован каркас распределенной системы мониторинга и диспетчеризации процессов гетерогенной среды, с применем современных подходов и технологий программирования распределенных систем.

Спроектированное и разработанное в рамках дипломного проекта программное обеспечение каркаса распределенной системы мониториннга может быть использованно для построения высоконагруженных систем мониторинга и дисетчеризациипроцессов на базе распределенных гетероегенных сетей.

В первую очередь проект ориентирован на использование в комерческом секторе для обеспечения отказоустойчивости серверов, рабочих станций и встраиваемых устройств, работающих под большой нагрузкой в режиме 24/7.

Можно выделить несколько путей развития проекта:

- разработка дополнительных модулей мониторинга для решений круга повседневных задач;
- реализация программного обеспечения службы мониторинга на нативном ЯП (например, на C++) с целью повышения быстродействия и надежности целевой системы;

- совершенствование компонентов и оптимизация алгоритмов базовой платформы службы мониторинга;
- полномасштабное внедрение и нагружочное тестирование системы на базе существующей инфраструктуры предприятия, например лаборатории МикроЭВМАлтГТУ;
- сопровождение системы, информационная и техническая поддержка пользователей и администраторов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Таненбаум, Э. Распределенные системы. Принципы и парадигмы / Э. Таненбаум, М. Ван Стейн. — СПб.: Питер, 2003. — 877 с: ил. — (Серия «Классика computerscience»).
2. Буч, Г. Объектно-ориентированный анализ и проектирование с примерами приложений / Г. Буч, Р. Максимчук, М. Энгл, Б. Янг, Д. Коналлен, К. Хьюстон – Вильямс, 2010. – 720 с: ил.
3. Homepage of Zabbix [Электронный ресурс]/ Режим доступа: <http://www.zabbix.com>
4. Ganglia Monitoring System [Электронный ресурс]/ Режим доступа: <http://ganglia.sourceforge.net>
5. Nagios [Электронный ресурс] / Режим доступа: <http://www.nagios.org>
6. Mon – Service Monitoring Daemon [Электронный ресурс] / Режим доступа: https://mon.wiki.kernel.org/index.php/Main_Page
7. Network Monitoring Software Tools with Big Brother by Quest Software [Электронный ресурс] / Режим доступа: <http://www.quest.com/big-brother>
8. Network Monitoring | Zenoss [Электронный ресурс] / Режим доступа: <http://www.zenoss.com/product/network>
9. Eclipse Project [Электронный ресурс] / Режим доступа: <http://www.eclipse.org/>
10. NetBeans Project [Электронный ресурс] / Режим доступа: <http://www.netbeans.org/>
11. Сайт IntelliJIDEA [Электронный ресурс] / Режим доступа: <http://www.jetbrains.com/idea/>
12. Apache log4j [Электронный ресурс] / Режим доступа: <http://logging.apache.org/log4j/>
13. Henning, Michi. Distributed Programming with Ice [Электронный ресурс]: / Michi Henning, Mark Spruiell; ZeroC, Inc., 2009. - Режим доступа: <http://zeroc.com/download/Ice/3.3/Ice-3.3.1.pdf>
14. Гамма, Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. – СПб.: Питер, 2009. – 366 с.
15. Сайт Java [Электронный ресурс] / Режим доступа: <http://java.com/en/>

16. Сайт Python [Электронный ресурс] / Режим доступа:
<http://www.python.org/>
17. Java Universal Network/Graph Framework. [Электронный ресурс] / Режим доступа: <http://jung.sourceforge.net/index.html>
18. Apache subversion [Электронный ресурс] / Режим доступа:
<http://subversion.apache.org/>
19. Олифер, В. Г. Компьютерные сети. Принципы, технологии, протоколы. Учебник для вузов / В.Г. Олифер, Н. А. Олифер. — СПб.: Питер, 2007. — 960 с.

Приложение А

Задание на дипломное проектирование

Приложение Б

Руководство администратора

Б.1 Организация среды исполнения

Для того чтобы запустить ядро на узле, необходимо предварительно развернуть систему исполнения. Для этого потребуется:

- а) установить виртуальную машину Java (JRE) не ниже версии 1.4;
- б) установить интерпретатор языка Python предпочтительной версии 2.6.

Требуемое программное обеспечение не является специфичным, а наоборот очень распространено. Поэтому никаких трудностей установка не доставит. К тому же никаких дополнительных настроек интерпретаторы не требуют. Единственное, что необходимо, это добавление путей до исполняемых файлов интерпретаторов в переменную системы PATH. Это необходимо чтобы интерпретаторы могли запускаться из любой директории в командной строке.

Б.2 Запуск ядра

Приложение ядра распространяется в виде бинарных файлов, что означает отсутствие необходимости установки. Достаточно скопировать архив на в ПЗУ узла и ядро готово к работе. Рекомендуемым вариантом эксплуатации, является запуск программы как сервиса windows, либо как программы демона в семействах unix. Запуск программ в фоновом режиме в unix системах осуществляется стандартными средствами операционной системы (за подробностями необходимо обратиться к справочной документации вашего дистрибутива).

В комплекте с бинарными файлами идут два скриптовых файла (рисунок Б.1). В них содержатся ключи для интерпретаторов и начальные настройки приложения. Первый файл это snoopyd.cmd используется в операционных семействах Windows. Второй – snoopyd.sh используется в операционных системах семейства unix.

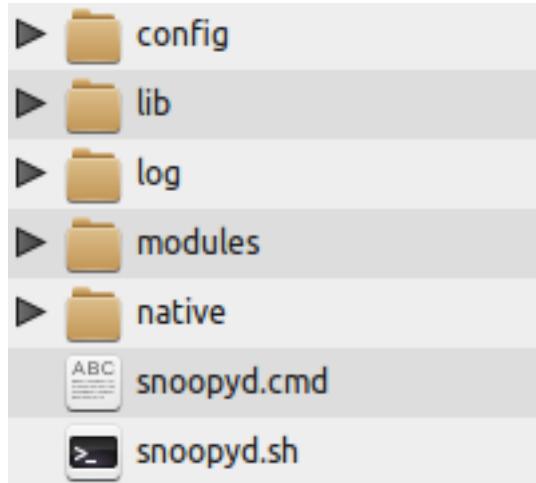


Рисунок Б.1 – Комплектация приложения ядра

Если во время запуска возникли проблемы, необходимо проверить:

- а) занесены ли пути до исполняемых файлов интерпретаторов в переменную среды PATH;
- б) сохранен ли файл в той кодировке, которая правильно декодируется в вашей операционной системе;
- в) проверить поддерживает ли заданные ключи версия интерпретаторов, установленных в вашей системе.
- г) проверить наличие конфигурационных файлов, указанных в скриптах.

После запуска ядра, если оно запущено как приложение, то в стандартный поток вывода консоли начнутся печатать сообщения ядра. О настройке подробности печатаемых сообщений необходимо обратиться к документации библиотеки log4j на официально сайте apache[link].

Запуск на одном узле нескольких ядер возможен, но не рекомендуется. Для взаимодействия используется NUID, который уникален для каждого ядра, но для отображения пользователю используется имя узла.

```
C:\snoopyd>snoopyd.cmd
com.googlecode.snoopyd.core.Snoopyd - running snoopyd 0.0.0
com.googlecode.snoopyd.core.Kernel - init kernel drivers
com.googlecode.snoopyd.core.Kernel - init kernel adapters
com.googlecode.snoopyd.core.Kernel - init primary ice adapter
com.googlecode.snoopyd.core.Kernel - primary adapter endpoints is a "tcp -h 192.168.0.10 -p 10000"
com.googlecode.snoopyd.core.Kernel - init secondary ice adapter
com.googlecode.snoopyd.core.Kernel - primary secondary endpoints is a "udp -h 192.168.0.10 -p 10000"
com.googlecode.snoopyd.core.Kernel - init kernel listeners
com.googlecode.snoopyd.core.Kernel - init kernel filters
com.googlecode.snoopyd.core.Kernel - init kernel rate
com.googlecode.snoopyd.core.Kernel - init self session
com.googlecode.snoopyd.core.Kernel - starting kernel thread
com.googlecode.snoopyd.core.Snoopyd - ... identity: dev/9dbaaa52-8902-4cd7-89b7-1252a6a247ca
com.googlecode.snoopyd.core.Snoopyd - ... hostname: Node3
com.googlecode.snoopyd.core.Snoopyd - ... rate: 1?
com.googlecode.snoopyd.core.Snoopyd - setting shutdown hook for snoopyd
com.googlecode.snoopyd.core.Snoopyd - fetching kernel drivers:
com.googlecode.snoopyd.core.Snoopyd - ... Sessionier
com.googlecode.snoopyd.core.Snoopyd - ... Modular
com.googlecode.snoopyd.core.Snoopyd - ... Aliver
com.googlecode.snoopyd.core.Snoopyd - ... Networker
com.googlecode.snoopyd.core.Snoopyd - ... Resulter
com.googlecode.snoopyd.core.Snoopyd - ... Configurer
com.googlecode.snoopyd.core.Snoopyd - ... Hoster
com.googlecode.snoopyd.core.Snoopyd - ... Scheduler
com.googlecode.snoopyd.core.Snoopyd - ... Controller
com.googlecode.snoopyd.core.Snoopyd - ... Discoverer
com.googlecode.snoopyd.core.Snoopyd - fetching drivers adapters:
com.googlecode.snoopyd.core.Snoopyd - ... DiscovererAdapter
com.googlecode.snoopyd.core.Snoopyd - ... SessionierAdapter
com.googlecode.snoopyd.core.Kernel - handle SnoopydStartedEvent with SuspenseHandler
com.googlecode.snoopyd.core.Kernel - init kernel
com.googlecode.snoopyd.core.Kernel - ... activating Networker
com.googlecode.snoopyd.core.Kernel - ... activating Scheduler
com.googlecode.snoopyd.core.Kernel - handle NetworkEnabledEvent with SuspenseHandler
com.googlecode.snoopyd.core.Kernel - handle KernelStateChangedEvent with SuspenseHandler
com.googlecode.snoopyd.core.Kernel - changing kernel state on OnlineState
com.googlecode.snoopyd.core.Kernel - handle DiscoverRecivedEvent with OnlineHandler
com.googlecode.snoopyd.core.Kernel - handle ChildSessionReceivedEvent with OnlineHandler
com.googlecode.snoopyd.core.Kernel - handle ParentSessionSendedEvent with OnlineHandler
com.googlecode.snoopyd.core.Kernel - handle ChildSessionSendedEvent with OnlineHandler
com.googlecode.snoopyd.core.Kernel - handle ParentSessionReceivedEvent with OnlineHandler
com.googlecode.snoopyd.core.Kernel - handle KernelStateChangedEvent with OnlineHandler
com.googlecode.snoopyd.core.Kernel - changing kernel state on ActiveState
com.googlecode.snoopyd.driver.Aliver - starting Aliver
com.googlecode.snoopyd.driver.Resulter - starting Resulter
com.googlecode.snoopyd.driver.Resulter - use connection url: null
com.googlecode.snoopyd.driver.Aliver - parent node is alive: dev/9dbaaa52-8902-4cd7-89b7-1252a6a247ca
com.googlecode.snoopyd.driver.Aliver - child node is alive: dev/9dbaaa52-8902-4cd7-89b7-1252a6a247ca
com.googlecode.snoopyd.driver.Resulter - The url cannot be null
com.googlecode.snoopyd.driver.Scheduler - starting Scheduler
com.googlecode.snoopyd.core.Kernel - handle ScheduleUpdatedEvent with ActiveHandler
com.googlecode.snoopyd.driver.Scheduler - synchronize shceduler with dev/9dbaaa52-8902-4cd7-89b7-1252a6a247ca
com.googlecode.snoopyd.driver.Scheduler - updating scheduler
com.googlecode.snoopyd.driver.Scheduler - set scheduler for node dev/9dbaaa52-8902-4cd7-89b7-1252a6a247ca
com.googlecode.snoopyd.driver.Scheduler - set schedule for module 53abc066-f972-470d-hebf-d598231fcc3
com.googlecode.snoopyd.driver.Scheduler - set schedule for module 0db484f3-9c9d-4423-87fe-a2e331d6fa66
com.googlecode.snoopyd.driver.Scheduler - set schedule for module d2545757-0011-4fed-ad55-072526b09279
com.googlecode.snoopyd.driver.Scheduler - set schedule for module 385425cf-ca5d-4c18-a9d1-1c841d701a3e
com.googlecode.snoopyd.driver.Scheduler - set schedule for module 7edc819d-ab97-432b-8c87-8cc8e8686fca
com.googlecode.snoopyd.core.Kernel - handle ScheduleTimeComeEvent with ActiveHandler
com.googlecode.snoopyd.core.Kernel - handle ScheduleTimeComeEvent with ActiveHandler
com.googlecode.snoopyd.core.Kernel - handle InvocationEvent with ActiveHandler
com.googlecode.snoopyd.core.Kernel - handle InvokationEvent with ActiveHandler
com.googlecode.snoopyd.core.Kernel - handle ResultRecieivedEvent with ActiveHandler
com.googlecode.snoopyd.driver.Resulter - storing result []
com.googlecode.snoopyd.core.Kernel - handle ResultRecieivedEvent with ActiveHandler
com.googlecode.snoopyd.driver.Resulter - storing result []
```

Рисунок Б.2 – Вывод сообщений ядра в консоль

Б.3 Работа с панелью управления

Структура главного окна панели довольно проста:

- дерево узлов;
- область служебных окн;
- карта узлов (Network Map);
- текстовый редактор (Notepad);

- д) свойства узлов и модулей (properties);
- е) результаты работы модулей (results).

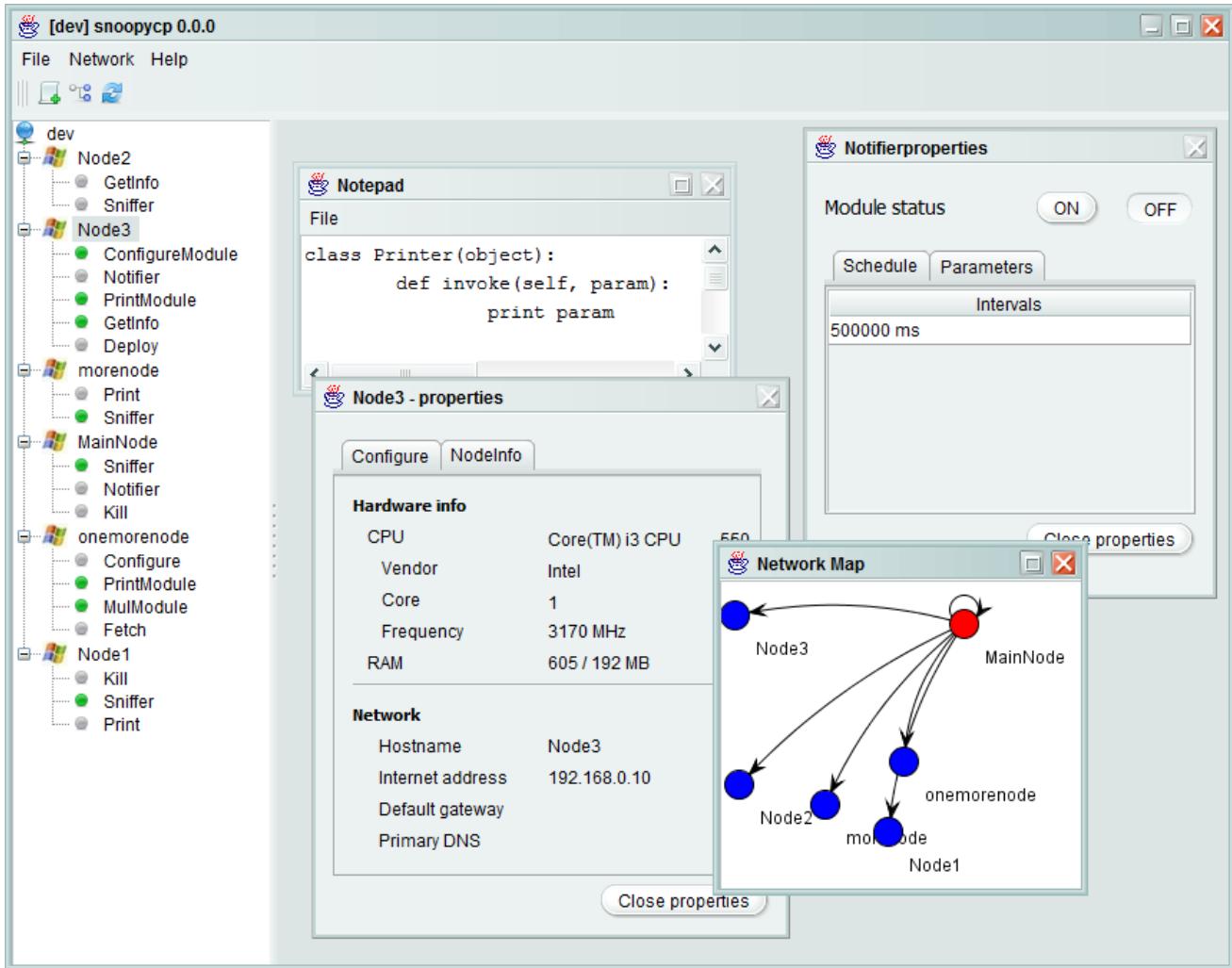


Рисунок Б.3 – Главное окно панели управления

Б.3.1 Запуск панели управления

Панель управления так же как и ядро распространяется в виде бинарных файлов. Но в отличии от ядра требует только наличия виртуальной машины Java. Это означает, что на любом узле, где запущено ядро, панель управления тоже запустится. В комплекте поставки так же имеются два исполняемых скрипта для ОС семейства windows и unix. При возникновении трудностей необходимо обратиться к пункту Б.2 Запуск ядра, в котором описаны некоторые способы устранение неполадок.

После запуска одного из стартовых скриптов появится главное окно (рисунок Б.3). Точки входа в сеть может быть любой узел сети или подсети. Если

узел отделен о ядра или ни одного ядра не запущено, то в главном окне ничего не будет отображено.

После запуска панели необходимо время для регистрации узлов у домене. Это может занять несколько секунд. В это время в главном окне может ничего не отображаться.

Б.3.2 Развёртывание модуля узла

Для того чтобы назначить выполнение какой-либо задачи узлу, достаточно:

- 1) во встроенным текстовом редакторе File -> Notepad написать исходный код модуля. Так же можно загрузить уже готовый исходный текст в редактор;

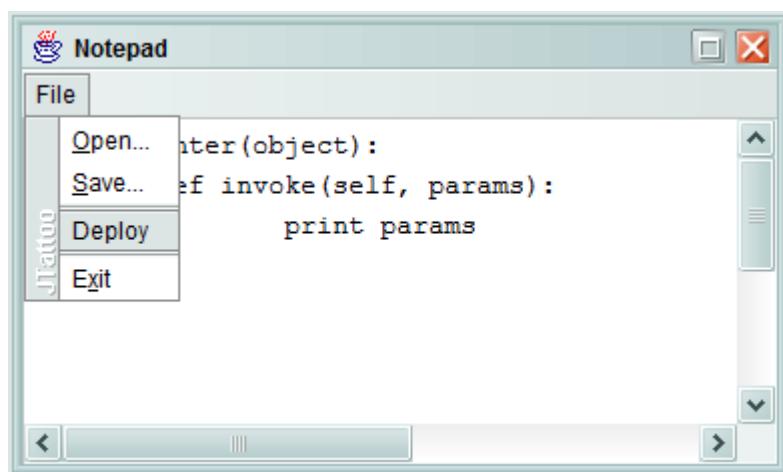


Рисунок Б.4 – Встроенный текстовый редактор

- 2) выбрать из списка доступных(Available) узлов те узлы, к которым необходимо прикрепить задачу, и перемещаем их в список узлов, на которые будет происходить развертывание(Deploied). Затем выставить начальные параметры;

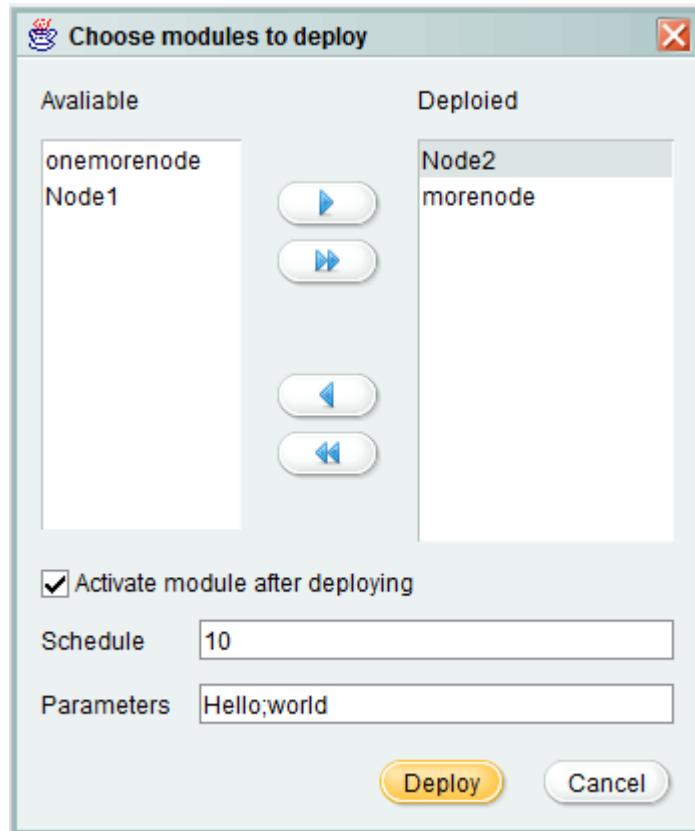


Рисунок Б.5 – Выбор узлов для развертывания модуля

- 3) после нажатия кнопки Deploy, ждать результатов выполнения задачи.

Б.3.3 Просмотр информации

Дерево узлов доступно сразу и не требует абсолютно никаких дополнительных действий со стороны пользователя. Оно обновляется автоматически.

Для просмотра карты узлов необходимо выбрать Network -> View full map. После этого появится отдельно окно с изображение графа.

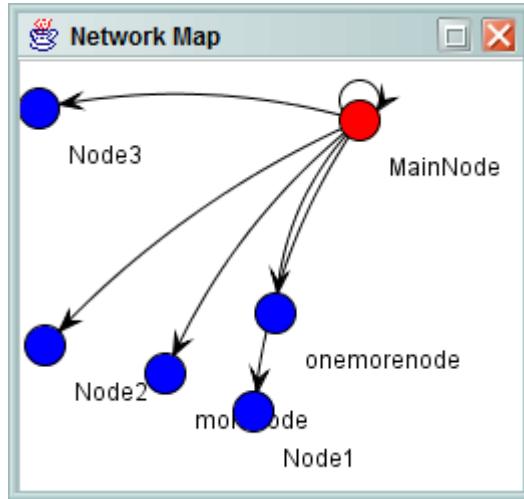


Рисунок Б.6 – Карта узлов

Для просмотра свойств узла или модуля, необходимо в соответствующем контекстном меню выбрать пункт Properties.

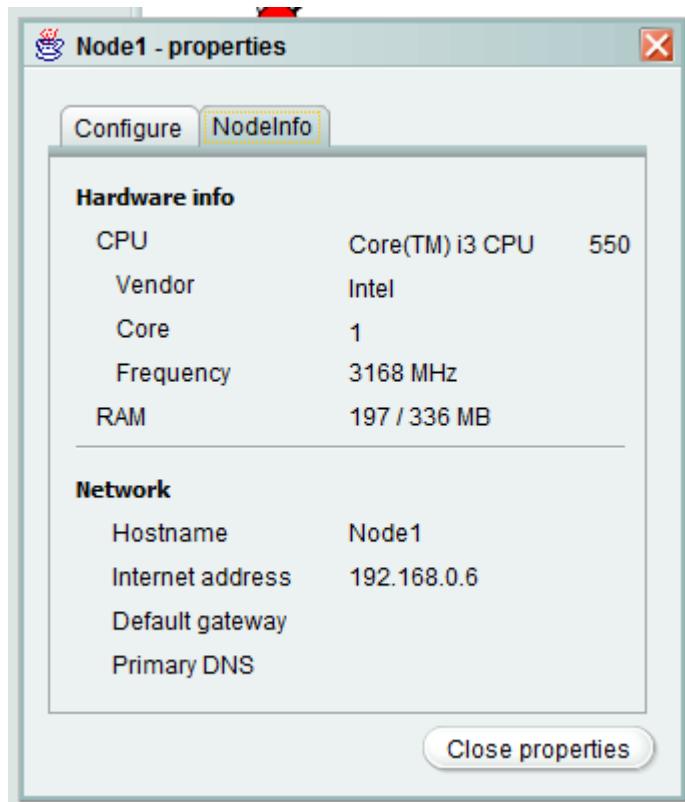


Рисунок Б.7 – Свойства модуля

Для просмотра результатов работы модулей необходимо выбрать в контекстном меню соответствующего узла или модуля пункта Results.

Рисунок Б.8 – Список результатов работы модулей

Приложение В

Код программы

<snoopyd.ice>

```
// ****
// Copyright 2011 Snoopy Project
//
// Licensed under the Apache License, Version 2.0 (the "License");
// You may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.
// ****

#ifndef SNOOPYD_ICE
#define SNOOPYD_ICE

#include <Ice/Identity.ice>

module com { module googlecode { module snoopyd {

    module driver
    {
        interface IHoster;
        interface IDiscoverer;
        interface ISessionier;
        interface IController;
        interface IModuler;
        interface IScheduler;
        interface IConfigurer;

    };

    module session
    {
        interface ISession
        {
            void refresh();
            void destroy();
        };

        interface IKernelSession extends ISession
        {
            driver::IModuler* moduler();
            driver::IScheduler* scheduler();
        };

        interface IUserSession extends ISession
        {
            driver::IHoster* hoster();
        };
    };
}}};
```

```

        driver::IController* controller();
        driver::IModuler* moduler();
        driver::IConfigurer* configurer();
        driver::IScheduler* scheduler();
    };

};

module driver
{
    exception ModuleNotFoundException
    {
    };

    dictionary<string, string> StringMap;
    sequence<string> StringArray;
    sequence<long> LongArray;

    interface IDiscoverer
    {
        void discover(Ice::Identity identity);
    };

    interface ISessionier
    {
        session::IKernelSession* createKernelSession(Ice::Identity identity,
session::IKernelSession* selfSession);
        session::IUserSession* createUserSession(Ice::Identity identity,
session::IUserSession* selfSession);
    };

    interface IHoster
    {
        StringMap context();
    };

    interface IController
    {
        void shutdown();
    };

    interface IModuler
    {
        StringMap fetch();
        void deploy(string muid, string code);
        void undeploy(string muid) throws ModuleNotFoundException;

        StringArray launch(string muid, StringArray params) throws
ModuleNotFoundException;

        void force(string muid, StringArray params);
    };

    interface IScheduler
    {
        void synchronize(Ice::Identity identity, IScheduler* remoteScheduler);

        StringMap timetable();
        StringMap statetable();
        StringMap paramtable();
    };
}

```

```

        void schedule(string muid, LongArray delays, StringArray params);
        void unschedule(string muid) throws ModuleNotFoundException;

        void force(Ice::Identity identity, string muid, StringArray params);

        void toggle(string muid);

    };

    interface IConfigurer
    {
        void reconfigure(StringMap configuration);
        StringMap configuration();
    };

};

};

};

#endif

```

<Launcher.java>

```

/*
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopyd;

import com.googlecode.snoopyd.core.Snoopyd;

public class Launcher {

    public static void main(String[] args) {
        Snoopyd snoopyd = new Snoopyd();

        int status = snoopyd.main(Defaults.APP_NAME, args,
                System.getProperty("snoopyd.configuration",
Defaults.CONFIGURATION));

        System.exit(status);
    }
}
```

<Defaults.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.snoopyd;  
  
public final class Defaults {  
  
    public static final String APP_NAME = "snoopyd";  
    public static final String APP_VER = "0.0.0";  
  
    public static final String CONFIGURATION = "snoopyd.conf";  
  
    public static final String PRIMARY_ADAPTER_NAME = "PrimaryAdapter";  
    public static final String SECONDARY_ADAPTER_NAME = "SecondaryAdapter";  
  
    public static final int DISCOVER_INTERVAL = 5000;  
    public static final int DISCOVER_TIMEOUT = 6 * DISCOVER_INTERVAL;  
  
    public static final int ALIVE_INTERVAL = 15000;  
  
    public static final int MODULER_INTERVAL = 15000;  
  
    public static final int NETWORKER_INTERVAL = 3000;  
  
    public static final int RESULTER_THRESHOLD = 5;  
  
    public static final String KERNEL_THREAD_NAME = "Kernel-Thread";  
    public static final String ALIVER_THREAD_NAME = "Aliver-Thread";  
    public static final String DISCOVERER_THREAD_NAME = "Discoverer-Thread";  
    public static final String INVOKER_THREAD_NAME = "Invoker-Thread";  
    public static final String MODULER_THREAD_NAME = "Moduler-Thread";  
    public static final String TIMER_THREAD_NAME = "Timer-Thread";  
    public static final String INVOKATION_THREAD_NAME = "Invokation-Thread";  
    public static final String RESULTER_THREAD_NAME = "Resulter-Thread";  
  
    /*  
     * This is rate from my first laptop (Samsung R20).  
     */  
    public static final int BASELINE_RATE = 912;  
}
```

<Snoopyd.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.snoopyd.core;  
  
import org.apache.log4j.Logger;  
  
import com.googlecode.snoopyd.Defaults;  
import com.googlecode.snoopyd.adapter.Adapter;  
import com.googlecode.snoopyd.core.event.KernelEvent;  
import com.googlecode.snoopyd.core.event.SnoopydStartedEvent;  
import com.googlecode.snoopyd.core.event.SnoopydTerminatedEvent;  
import com.googlecode.snoopyd.driver.Driver;  
import com.googlecode.snoopyd.util.Identities;  
  
public class Snoopyd extends Ice.Application {  
  
    private static Logger logger = Logger.getLogger(Snoopyd.class);  
  
    public static final int EXIT_SUCCESS = 0;  
    public static final int EXIT_FAILURE = 999;  
  
    public static class ShutdownHook extends Thread {  
  
        private Kernel kernel;  
  
        public ShutdownHook(Kernel kernel) {  
            this.kernel = kernel;  
        }  
  
        @Override  
        public void run() {  
  
            KernelEvent event = new SnoopydTerminatedEvent();  
            kernel.handle(event);  
  
            synchronized (event) {  
                try {  
                    event.wait();  
                } catch (InterruptedException e) {  
                    logger.debug(e.getMessage());  
                }  
            }  
        }  
    }  
}
```

```

@Override
public int run(String[] args) {
    logger.info("running " + Defaults.APP_NAME + " " + Defaults.APP_VER);

    Kernel kernel = new Kernel(communicator());

    logger.debug("... identity: " + Identities.toString(kernel.identity()));
    logger.debug("... hostname: " + kernel.hostname());
    logger.debug("... rate: " + kernel.rate());

    logger.debug("setting shutdown hook for snoopyd");
    setInterruptHook(new ShutdownHook(kernel));

    logger.info("fetching kernel drivers:");
    for (Driver drv : kernel.drivers()) {
        logger.debug("... " + drv.name());
    }

    logger.info("fetching drivers adapters: ");
    for (Adapter adp : kernel.adapters()) {
        logger.debug("... " + adp.name());
    }

    kernel.handle(new SnoopydStartedEvent());
    kernel.waitForTerminated();

    return EXIT_SUCCESS;
}
}

```

<Kernel.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopyd.core;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.Collection;
import java.util.Collections;
import java.util.HashMap;
import java.util.LinkedList;

```

```

import java.util.List;
import java.util.Map;
import java.util.Properties;
import java.util.Queue;

import org.apache.log4j.Logger;

import Ice.Identity;

import com.googlecode.snoopyd.Defaults;
import com.googlecode.snoopyd.adapter.Adapter;
import com.googlecode.snoopyd.adapter.DiscovererAdapter;
import com.googlecode.snoopyd.adapter.SessionierAdapter;
import com.googlecode.snoopyd.core.event.KernelEvent;
import com.googlecode.snoopyd.core.filter.KernelFilter;
import com.googlecode.snoopyd.core.filter.KernelFilter.FilterAction;
import com.googlecode.snoopyd.core.filter.ToogleFilter;
import com.googlecode.snoopyd.core.state.KernelListener;
import com.googlecode.snoopyd.core.state.KernelState;
import com.googlecode.snoopyd.core.state.SuspenseState;
import com.googlecode.snoopyd.driver.Activable;
import com.googlecode.snoopyd.driver.Aliver;
import com.googlecode.snoopyd.driver.Configurer;
import com.googlecode.snoopyd.driver.Controller;
import com.googlecode.snoopyd.driver.Discoverer;
import com.googlecode.snoopyd.driver.Driver;
import com.googlecode.snoopyd.driver.Hoster;
import com.googlecode.snoopyd.driver.Moduler;
import com.googlecode.snoopyd.driver.Networker;
import com.googlecode.snoopyd.driver.Resulter;
import com.googlecode.snoopyd.driver.Scheduler;
import com.googlecode.snoopyd.driver.Sessionier;
import com.googlecode.snoopyd.driver.Startable;
import com.googlecode.snoopyd.session.IKernelSessionPrx;
import com.googlecode.snoopyd.session.IKernelSessionPrxHelper;
import com.googlecode.snoopyd.session.ISessionPrx;
import com.googlecode.snoopyd.session.KernelSession;
import com.googlecode.snoopyd.session.KernelSessionAdapter;
import com.googlecode.snoopyd.util.Identities;
import com.googlecode.snoopymm.IModuleManagerPrx;
import com.googlecode.snoopymm.IModuleManagerPrxHelper;

public class Kernel implements Runnable {

    public static class KernelException extends RuntimeException {

        public KernelException() {
            super();
        }

        public KernelException(String msg) {
            super(msg);
        }
    }

    public static Logger logger = Logger.getLogger(Kernel.class);

    private Identity identity;

    private Ice.Communicator communicator;
    private Ice.Properties properties;

```

```

private Ice.ObjectAdapter primary;
private Ice.ObjectAdapter secondary;

private Thread self;

private KernelState state;

private int rate;

private Properties configuration;

private Queue<KernelEvent> pool;

private HashMap<Class<?>, Driver> drivers;
private HashMap<Class<?>, Adapter> adapters;

private Map<Ice.Identity, ISessionPrx> parents;
private Map<Ice.Identity, ISessionPrx> childs;

private Map<Ice.Identity, Map<String, String>> cache;

private List<KernelListener> kernelListeners;
private List<KernelFilter> kernelFilters;

private IModuleManagerPrx moduleManager;

private IKernelSessionPrx selfSession;

private boolean started;

public Kernel(Ice.Communicator communicator) throws KernelException {

    this.rate = Integer.MIN_VALUE;

    this.pool = new LinkedList<KernelEvent>();

    this.cache = new HashMap<Identity, Map<String, String>>();

    this.parents = new HashMap<Identity, ISessionPrx>();
    this.childs = new HashMap<Identity, ISessionPrx>();

    this.state = new SuspenseState(this);

    this.communicator = communicator;
    this.properties = communicator.getProperties();

    this.identity = Identities.randomIdentity(properties
        .getProperty("Snoopy.Domain"));

    logger.debug("init kernel drivers");
    initDrivers();

    logger.debug("init kernel adapters");
    initAdapters();

    logger.debug("init primary ice adapter");
    initPrimaryAdapter();
    logger.info("primary adapter endpoints is a \""
        + primaryPublishedEndpoints() + "\"");
}

```

```

        logger.debug("init secondary ice adapter");
        initSecondaryAdapter();

        logger.debug("primary secondary endpoints is a \""
                     + secondaryPublishedEndpoints() + "\"");

        logger.debug("init kernel listeners");
        initKernelListeners();

        logger.debug("init kernel filters");
        initKernelFilters();

        logger.debug("init kernel rate");
        initKernelRate();

        logger.debug("init self session");
        initSelfSession();

        logger.debug("starting kernel thread");
        self = new Thread(this, Defaults.KERNEL_THREAD_NAME);
        self.start();
        started = true;
    }

    public String hostname() {
        try {
            return InetAddress.getLocalHost().getHostName();
        } catch (UnknownHostException ex) {
            return "localhost";
        }
    }

    public String os() {
        return System.getProperty("os.name");
    }

    public Properties configuration() {
        return configuration;
    }

    public void reconfigure(Map<String, String> configuration) {
        checkKernelThread();

        for (String key: configuration.keySet()) {
            this.configuration.put(key, configuration.get(key));
        }
    }

    public Identity identity() {
        return identity;
    }

    public int rate() {
        return rate;
    }

    public IKernelSessionPrx self() {
        return selfSession;
    }
}

```

```

public String primaryPublishedEndpoints() {
    return primary.getPublishedEndpoints()[primary.getPublishedEndpoints().length
- 1]
        ._toString();
}

public String secondaryPublishedEndpoints() {
    return secondary.getPublishedEndpoints()[secondary
        .getPublishedEndpoints().length - 1]._toString();
}

public String primaryEndpoints() {
    return primary.getEndpoints()[primary.getEndpoints().length - 1]
        ._toString();
}

public String secondaryEndpoints() {
    return secondary.getEndpoints()[secondary.getEndpoints().length - 1]
        ._toString();
}

public Ice.Communicator communicator() {
    return communicator;
}

public Ice.Properties properties() {
    return properties;
}

public Ice.ObjectAdapter primary() {
    return primary;
}

public Ice.ObjectAdapter secondary() {
    return secondary;
}

public KernelState state() {
    return state;
}

public IModuleManagerPrx moduleManager() {
    return moduleManager;
}

public void proccess(Throwable exception) {
    logger.error(exception.getMessage());
}

public void toogle(KernelState kernelState) {

    checkKernelThread();

    if (state.getClass() != kernelState.getClass()) {

        logger.info("changing kernel state on "
            + kernelState.getClass().getSimpleName());

        state = kernelState;
    }
}

```

```

        for (KernelListener listener : kernelListeners) {
            listener.stateChanged(kernelState);
        }
    }

public void init() {

    checkKernelThread();

    logger.debug("init kernel");

    for (Driver drv : drivers.values()) {
        if (drv instanceof Activable) {
            logger.debug("... activating " + drv.name());
            ((Activable) drv).activate();
        }
    }

    try {

        moduleManager = IModuleManagerPrxHelper.checkedCast(communicator
            .propertyToProxy("ModuleManager.Proxy"));

    } catch (Ice.ConnectionRefusedException ex) {
        throw new KernelException("could not connect to module manager");
    }

    loadConfiguration();

    primary.activate();
    secondary.activate();
}

public void dispose() {

    checkKernelThread();

    logger.debug("dispose kernel");

    started = false;

    for (Driver drv : drivers.values()) {
        if (drv instanceof Activable) {
            logger.debug("... deactivating " + drv.name());
            ((Activable) drv).deactivate();
        }
    }

    for (Driver drv : drivers.values()) {
        if (drv instanceof Startable) {
            if (((Startable) drv).started()) {
                logger.debug("... stopping " + drv.name());
                ((Startable) drv).stop();
            }
        }
    }

    saveConfiguration();
}

```

```

primary.deactivate();
secondary.deactivate();

        self.interrupt();
    }

public void waitForTerminated() {

    try {
        self.join();
    } catch (InterruptedException ignored) {
    }

}

@Override
public void run() {

    try {

        for (;started; ) {

            for ( ; !pool.isEmpty() && started; ) {

                KernelEvent event = pool.poll();

                boolean eventFiltered = false;
                KernelFilter usedFilter = null;

                for (KernelFilter filter : kernelFilters) {
                    if (FilterAction.REJECT == filter.accept(event)) {
                        eventFiltered = true;
                        usedFilter = filter;
                        break;
                    }
                }

                if (eventFiltered) {
                    logger.debug("filter " + event.name() + " with "
+
usedFilter.getClass().getSimpleName());
                } else {
                    logger.debug("handle " + event.name() + " with "
+
state.handler().getClass().getSimpleName());
                        state.handler().handle(event);
                }
            }

            synchronized (this) {
                try {
                    wait();
                } catch (InterruptedException ignored) {
                }
            }
        }

    } catch (KernelException ex) {
        logger.error(ex.getMessage());
        dispose();
    }
}

```

```

        }

    }

    public synchronized void handle(KernelEvent event) {
        pool.offer(event);
        notify();
    }

    public KernelEvent peek() {
        return pool.peek();
    }

    public Driver driver(Class<?> clazz) {
        return drivers.get(clazz);
    }

    public Collection<Driver> drivers() {
        return Collections.unmodifiableCollection(drivers.values());
    }

    public Adapter adapter(Class<?> clazz) {
        return adapters.get(clazz);
    }

    public Collection<Adapter> adapters() {
        return Collections.unmodifiableCollection(adapters.values());
    }

    public Collection<KernelListener> listeners() {
        return Collections.unmodifiableCollection(kernelListeners);
    }

    public Map<Ice.Identity, Map<String, String>> cache() {
        return cache;
    }

    public Map<Ice.Identity, ISessionPrx> parents() {
        return parents;
    }

    public Map<Ice.Identity, ISessionPrx> childs() {
        return childs;
    }

    private void checkKernelThread() {
        if (Thread.currentThread() != self) {
            throw new KernelException(
                "only kernel thread can change kernel state");
        }
    }

    private void initDrivers() {

        drivers = new HashMap<Class<?>, Driver>();

        drivers.put(Sessionier.class, new Sessionier(this));
        drivers.put(Discoverer.class, new Discoverer(this));
        drivers.put(Aliver.class, new Aliver(this));
        drivers.put(Networker.class, new Networker(this));
        drivers.put(Hoster.class, new Hoster(this));
    }
}

```

```

        drivers.put(Controller.class, new Controller(this));
        drivers.put(Scheduler.class, new Scheduler(this));
        drivers.put(Resulter.class, new Resulter(this));
        drivers.put(Moduler.class, new Moduler(this));
        drivers.put(Configurer.class, new Configurer(this));

    }

private void initAdapters() {
    adapters = new HashMap<Class<?>, Adapter>();
    adapters.put(
        DiscovererAdapter.class,
        new DiscovererAdapter(Identities
            .stringToIdentity(Discoverer.class.getSimpleName(),
                (Discoverer) drivers.get(Discoverer.class)));
    )
    adapters.put(SessionierAdapter.class, new SessionierAdapter(identity,
        (Sessionier) drivers.get(Sessionier.class)));
}

private void initPrimaryAdapter() {
    primary = communicator
        .createObjectAdapter(Defaults.PRIMARY_ADAPTER_NAME);

    primary.add((Ice.Object) adapters.get(SessionierAdapter.class),
        ((Adapter) adapters.get(SessionierAdapter.class)).identity());
}

private void initSecondaryAdapter() {
    secondary = communicator
        .createObjectAdapter(Defaults.SECONDARY_ADAPTER_NAME);

    secondary.add((Ice.Object) adapters.get(DiscovererAdapter.class),
        ((Adapter) adapters.get(DiscovererAdapter.class)).identity());
}

private void initKernelListeners() {
    kernelListeners = new LinkedList<KernelListener>();

    for (Driver driver : drivers.values()) {
        if (driver instanceof KernelListener) {
            kernelListeners.add((KernelListener) driver);
        }
    }
}

private void initKernelFilters() {
    kernelFilters = new LinkedList<KernelFilter>();

    kernelFilters.add(new ToogleFilter(this));
}

private void initKernelRate() {
    Hoster hoster = (Hoster) drivers.get(Hoster.class);
    Map<String, String> context = hoster.context();

    int ram = Integer.parseInt(context.get("Ram"));
    int mhz = Integer.parseInt(context.get("Mhz"));
}

```

```

        rate = (int) (((ram * 0.5 + mhz * 0.5) / Defaults.BASELINE_RATE) * 10);
    }

    private void initSelfSession() {
        selfSession = IKernelSessionPrxHelper
            .uncheckedCast(primary()).addWithUUID(
                new KernelSessionAdapter(new KernelSession(this))));
    }

    private void loadConfiguration() {
        try {

            configuration = new Properties();
            configuration.load(new
File(properties().getProperty("Snoopy.KernelConfiguration")));
        } catch (FileNotFoundException ex) {
        } catch (IOException ex) {
        }
    }

    private void saveConfiguration() {
        try {
            configuration.store(new
File(properties().getProperty("Snoopy.KernelConfiguration")),
"Snoopy.KernelConfiguration");
        } catch (FileNotFoundException ex) {
        } catch (IOException ex) {
        } catch (NullPointerException ex) {
        }
    }
}

```

<KernelState.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

```

```

package com.googlecode.snoopyd.core.state;

import com.googlecode.snoopyd.core.handler.KernelHandler;

public interface KernelState {
    public KernelHandler handler();
}

```

<KernelListener.java>

```
package com.googlecode.snoopyd.core.state;

public interface KernelListener {

    public void stateChanged(KernelState currentState);

}
```

<ActiveState.java>

```
/***
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
```

```
package com.googlecode.snoopyd.core.state;

import com.googlecode.snoopyd.core.Kernel;
import com.googlecode.snoopyd.core.handler.ActiveHandler;
import com.googlecode.snoopyd.core.handler.KernelHandler;

public class ActiveState implements KernelState {

    private Kernel kernel;

    private KernelHandler handler;

    public ActiveState(Kernel kernel) {
        this.kernel = kernel;
        this.handler = null;
    }

    @Override
    public KernelHandler handler() {
        if (handler == null) {
            synchronized (this) {
                handler = new ActiveHandler(kernel);
            }
        }
        return handler;
    }
}
```

<OfflineState.java>

```
/***
 * Copyright 2011 Snoopy Project
 *
```

```

* Licensed under the Apache License, Version 2.0 (the "License");
* You may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*      http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

```

```

package com.googlecode.snoopyd.core.state;

import com.googlecode.snoopyd.core.Kernel;
import com.googlecode.snoopyd.core.handler.KernelHandler;
import com.googlecode.snoopyd.core.handler.OfflineHandler;

public class OfflineState implements KernelState {

    private Kernel kernel;
    private KernelHandler handler;

    public OfflineState(Kernel kernel) {
        this.kernel = kernel;
        this.handler = null;
    }

    @Override
    public KernelHandler handler() {
        if (handler == null) {
            synchronized (this) {
                handler = new OfflineHandler(kernel);
            }
        }
        return handler;
    }
}

```

<OnlineState.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
*/

```

```

package com.googlecode.snoopyd.core.state;

import com.googlecode.snoopyd.core.Kernel;

```

```

import com.googlecode.snoopyd.core.handler.KernelHandler;
import com.googlecode.snoopyd.core.handler.OnlineHandler;

public class OnlineState implements KernelState {
    private Kernel kernel;
    private KernelHandler handler;

    public OnlineState(Kernel kernel) {
        this.kernel = kernel;
        this.handler = null;
    }

    @Override
    public KernelHandler handler() {
        if (handler == null) {
            synchronized (this) {
                handler = new OnlineHandler(kernel);
            }
        }
        return handler;
    }
}

```

<PassiveState.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

```

```

package com.googlecode.snoopyd.core.state;

import com.googlecode.snoopyd.core.Kernel;
import com.googlecode.snoopyd.core.handler.KernelHandler;
import com.googlecode.snoopyd.core.handler.PassiveHandler;

public class PassiveState implements KernelState {

    private Kernel kernel;

    private KernelHandler handler;

    public PassiveState(Kernel kernel) {
        this.kernel = kernel;
        this.handler = null;
    }

    @Override
    public KernelHandler handler() {
        if (handler == null) {

```

```

        synchronized (this) {
            handler = new PassiveHandler(kernel);
        }
    }
    return handler;
}
}

```

<SuspenseState.java>

```

/*
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
```

```

package com.googlecode.snoopyd.core.state;

import com.googlecode.snoopyd.core.Kernel;
import com.googlecode.snoopyd.core.handler.KernelHandler;
import com.googlecode.snoopyd.core.handler.SuspenseHandler;

public class SuspenseState implements KernelState {

    private Kernel kernel;

    private KernelHandler handler;

    public SuspenseState(Kernel kernel) {
        this.kernel = kernel;
        this.handler = null;
    }

    @Override
    public KernelHandler handler() {
        if (handler == null) {
            synchronized (this) {
                handler = new SuspenseHandler(kernel);
            }
        }
        return handler;
    }
}
```

<AbstractHandler.java>

```

/*
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
```

```

* You may obtain a copy of the License at
*
*      http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
package com.googlecode.snoopyd.core.handler;

import org.apache.log4j.Logger;

import com.googlecode.snoopyd.Defaults;
import com.googlecode.snoopyd.core.Kernel;
import com.googlecode.snoopyd.core.KernelException;
import com.googlecode.snoopyd.core.event.ChildNodeDeadedEvent;
import com.googlecode.snoopyd.core.event.ChildSessionRecivedEvent;
import com.googlecode.snoopyd.core.event.ChildSessionSendedEvent;
import com.googlecode.snoopyd.core.event.DiscoverRecivedEvent;
import com.googlecode.snoopyd.core.event.ExceptionEvent;
import com.googlecode.snoopyd.core.event.ForceStartEvent;
import com.googlecode.snoopyd.core.event.InvokationEvent;
import com.googlecode.snoopyd.core.event.KernelEvent;
import com.googlecode.snoopyd.core.event.KernelReconfiguredEvent;
import com.googlecode.snoopyd.core.event.KernelStateChangedEvent;
import com.googlecode.snoopyd.core.event.NetworkDisabledEvent;
import com.googlecode.snoopyd.core.event.NetworkEnabledEvent;
import com.googlecode.snoopyd.core.event.ParentNodeDeadedEvent;
import com.googlecode.snoopyd.core.event.ParentSessionRecivedEvent;
import com.googlecode.snoopyd.core.event.ParentSessionSendedEvent;
import com.googlecode.snoopyd.core.event.ResultRecievedEvent;
import com.googlecode.snoopyd.core.event.ScheduleTimeComeEvent;
import com.googlecode.snoopyd.core.event.ScheduleUpdatedEvent;
import com.googlecode.snoopyd.core.event.SnoopydStartedEvent;
import com.googlecode.snoopyd.core.event.SnoopydTerminatedEvent;
import com.googlecode.snoopyd.driver.ISchedulerPrx;
import com.googlecode.snoopyd.session.IKernelSessionPrx;

public abstract class AbstractHandler implements KernelHandler {

    private static Logger logger = Logger.getLogger(AbstractHandler.class);

    protected Kernel kernel;

    public AbstractHandler(Kernel kernel) {
        this.kernel = kernel;
    }

    @Override
    public void handle(KernelEvent event) {

        if (event instanceof NetworkEnabledEvent) {

            handle((NetworkEnabledEvent) event);

        } else if (event instanceof NetworkDisabledEvent) {

            handle((NetworkDisabledEvent) event);

```

```

} else if (event instanceof ChildSessionSendedEvent) {
    handle((ChildSessionSendedEvent) event);
} else if (event instanceof ChildSessionRecivedEvent) {
    handle((ChildSessionRecivedEvent) event);
} else if (event instanceof DiscoverRecivedEvent) {
    handle((DiscoverRecivedEvent) event);
} else if (event instanceof ParentNodeDeadedEvent) {
    handle((ParentNodeDeadedEvent) event);
} else if (event instanceof ChildNodeDeadedEvent) {
    handle((ChildNodeDeadedEvent) event);
} else if (event instanceof SnoopydStartedEvent) {
    handle((SnoopydStartedEvent) event);
} else if (event instanceof SnoopydTerminatedEvent) {
    handle((SnoopydTerminatedEvent) event);
} else if (event instanceof KernelStateChangedEvent) {
    handle((KernelStateChangedEvent) event);
} else if (event instanceof InvokationEvent) {
    handle((InvokationEvent) event);
} else if (event instanceof ScheduleTimeComeEvent) {
    handle((ScheduleTimeComeEvent) event);
} else if (event instanceof ExceptionEvent) {
    handle((ExceptionEvent) event);
} else if (event instanceof ScheduleUpdatedEvent) {
    handle((ScheduleUpdatedEvent) event);
} else if (event instanceof ParentSessionRecivedEvent) {
    handle((ParentSessionRecivedEvent) event);
} else if (event instanceof ParentSessionSendedEvent) {
    handle((ParentSessionSendedEvent) event);
} else if (event instanceof ResultRecievedEvent) {
    handle((ResultRecievedEvent) event);
}

```

```

        } else if (event instanceof ForceStartEvent) {
            handle((ForceStartEvent) event);
        } else if (event instanceof KernelReconfiguredEvent) {
            handle((KernelReconfiguredEvent) event);
        } else {
            logger.warn("not found handler for " + event.name());
        }
    }

    @Override
    public void handle(KernelReconfiguredEvent event) {
        kernel.reconfigure(event.configuration());
    }

    @Override
    public void handle(ForceStartEvent event) {
        for (Ice.Identity identity: kernel.parents().keySet()) {
            IKernelSessionPrx parentSession = (IKernelSessionPrx)
kernel.parents().get(identity);
            ISchedulerPrx parentScheduler = parentSession.scheduler();
            parentScheduler.force(kernel.identity(), event.muid(),
event.params());
        }
    }

    @Override
    public void handle(ResultRecievedEvent event) {
    }

    @Override
    public void handle(ScheduleUpdatedEvent event) {
        for (Ice.Identity identity: kernel.parents().keySet()) {
            try {
                IKernelSessionPrx parentSession = (IKernelSessionPrx)
kernel.parents().get(identity);
                ISchedulerPrx parentScheduler = parentSession.scheduler();
                parentScheduler.synchronize(kernel.identity(),
kernel.self().scheduler());
            } catch (Exception ex) {
            }
        }
    }

    @Override

```

```

public void handle(ParentSessionReceivedEvent event) {
    kernel.parents().put(event.identity(), event.session());
}

@Override
public void handle(ParentSessionSendedEvent event) {

}

@Override
public void handle(ChildSessionSendedEvent event) {

}

@Override
public void handle(ChildSessionReceivedEvent event) {
    kernel.childs().put(event.identity(), event.session());
}

@Override
public void handle(ExceptionEvent event) {
    if (event.exception() instanceof KernelException) {
        throw (KernelException) event.exception();
    } else {
        kernel.process(event.exception());
    }
}

@Override
public void handle(KernelStateChangedEvent event) {
    kernel.toggle(event.state());
}

@Override
public void handle(InvocationEvent event) {
    Thread thread = new Thread(event, Defaults.INVOKATION_THREAD_NAME + "-" +
java.util.UUID.randomUUID().toString());
    thread.start();
}

@Override
public void handle(SnoopydStartedEvent event) {
    kernel.init();
}

@Override
public void handle(SnoopydTerminatedEvent event) {
    kernel.dispose();

    synchronized (event) {
        event.notify();
    }
}
}

```

<ActiveHandler.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *      http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.snoopyd.core.handler;  
  
import Ice.ConnectionRefusedException;  
  
import com.googlecode.snoopyd.core.Kernel;  
import com.googlecode.snoopyd.core.event.ChildNodeDeadedEvent;  
import com.googlecode.snoopyd.core.event.DiscoverRecivedEvent;  
import com.googlecode.snoopyd.core.event.InvokationEvent;  
import com.googlecode.snoopyd.core.event.KernelStateChangedEvent;  
import com.googlecode.snoopyd.core.event.NetworkDisabledEvent;  
import com.googlecode.snoopyd.core.event.NetworkEnabledEvent;  
import com.googlecode.snoopyd.core.event.ParentNodeDeadedEvent;  
import com.googlecode.snoopyd.core.event.ResultRecievedEvent;  
import com.googlecode.snoopyd.core.event.ScheduleTimeComeEvent;  
import com.googlecode.snoopyd.core.state.OfflineState;  
import com.googlecode.snoopyd.core.state.OnlineState;  
import com.googlecode.snoopyd.driver.IModulerPrx;  
import com.googlecode.snoopyd.driver.ISessionierPrx;  
import com.googlecode.snoopyd.driver.ISessionierPrxHelper;  
import com.googlecode.snoopyd.driver.Resulter;  
import com.googlecode.snoopyd.driver.Scheduler;  
import com.googlecode.snoopyd.session.IKernelSessionPrx;  
import com.googlecode.snoopyd.session.IKernelSessionPrxHelper;  
import com.googlecode.snoopyd.session.KernelSession;  
import com.googlecode.snoopyd.session.KernelSessionAdapter;  
import com.googlecode.snoopyd.util.Identities;  
  
public class ActiveHandler extends AbstractHandler implements KernelHandler {  
  
    public ActiveHandler(Kernel kernel) {  
        super(kernel);  
    }  
  
    @Override  
    public void handle(NetworkEnabledEvent event) {  
  
    }  
  
    @Override  
    public void handle(NetworkDisabledEvent event) {  
  
        kernel.childs().clear();  
        kernel.parents().clear();  
    }  
}
```

```

        String proxy = Identities.toString(kernel.identity()) + ":" +
                      kernel.primaryEndpoints();

        ISessionierPrx prx = ISessionierPrxHelper.checkedCast(kernel
            .communicator().stringToProxy(proxy));

        IKernelSessionPrx selfSession = IKernelSessionPrxHelper
            .uncheckedCast(kernel.primary()).addWithUUID(
                new KernelSessionAdapter(new
                    KernelSession(kernel))));

        IKernelSessionPrx remoteSession = prx.createKernelSession(
            kernel.identity(), selfSession);

        kernel.parents().put(kernel.identity(), remoteSession);

        kernel.handle(new KernelStateChangedEvent(new OfflineState(kernel)));
    }

    @Override
    public void handle(DiscoverReceivedEvent event) {
        kernel.cache().put(event.identity(), event.context());
    }

    @Override
    public void handle(ParentNodeDeadedEvent event) {

        kernel.cache().clear();
        kernel.parents().remove(event.identity());
        kernel.handle(new KernelStateChangedEvent(new OnlineState(kernel)));
    }

    @Override
    public void handle(ChildNodeDeadedEvent event) {

        kernel.childs().remove(event.identity());
        Scheduler scheduler = (Scheduler) kernel.driver(Scheduler.class);
        scheduler.cancel(event.identity());
    }

    @Override
    public void handle(ScheduleTimeComeEvent event) {

        final ScheduleTimeComeEvent fevent = event;

        kernel.handle(new InvokationEvent() {

            @Override
            public void run() {
                try {

                    IKernelSessionPrx session = (IKernelSessionPrx) kernel
                        .childs().get(fevent.identity());
                    IModulerPrx moduler = session.moduler();

                    try {
                        String result[] = moduler.launch(fevent.muid(),
                            fevent.params());
                        kernel.handle(new ResultRecievedEvent(

```

```

        fevent.identity(),      fevent.muid(),
result));

        }

(com.googlecode.snoopyd.driver.ModuleNotFoundException ex) {

        }

    } catch (ConnectionRefusedException ex) {
        // kernel.handle(new ExceptionEvent(new
        // KernelException("could not connect to module
manager")));
    } catch (Exception ex) {

        }

    }

});

}

@Override
public void handle(ResultReceivedEvent event) {
    super.handle(event);

    Resulter resulter = (Resulter) kernel.driver(Resulter.class);

    String hostname = kernel.cache().get(event.identity()).get("hostname");
    String osname = kernel.cache().get(event.identity()).get("os");

    IKernelSessionPrx remoteSession = (IKernelSessionPrx) kernel.childs()
        .get(event.identity());

    String module = remoteSession.moduler().fetch().get(event.muid());

    resulter.store(hostname, osname, module, event.result());
}
}

```

<KernelHandler.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

```

```

package com.googlecode.snoopyd.core.handler;

import com.googlecode.snoopyd.core.event.ChildNodeDeadedEvent;
import com.googlecode.snoopyd.core.event.ChildSessionReceivedEvent;
import com.googlecode.snoopyd.core.event.ChildSessionSendedEvent;
import com.googlecode.snoopyd.core.event.DiscoverReceivedEvent;

```

```

import com.googlecode.snoopyd.core.event.ExceptionEvent;
import com.googlecode.snoopyd.core.event.ForceStartEvent;
import com.googlecode.snoopyd.core.event.InvokationEvent;
import com.googlecode.snoopyd.core.event.KernelEvent;
import com.googlecode.snoopyd.core.event.KernelReconfiguredEvent;
import com.googlecode.snoopyd.core.event.KernelStateChangedEvent;
import com.googlecode.snoopyd.core.event.NetworkDisabledEvent;
import com.googlecode.snoopyd.core.event.NetworkEnabledEvent;
import com.googlecode.snoopyd.core.event.ParentNodeDeadedEvent;
import com.googlecode.snoopyd.core.event.ParentSessionRecivedEvent;
import com.googlecode.snoopyd.core.event.ParentSessionSendedEvent;
import com.googlecode.snoopyd.core.event.ResultRecievedEvent;
import com.googlecode.snoopyd.core.event.ScheduleTimeComeEvent;
import com.googlecode.snoopyd.core.event.ScheduleUpdatedEvent;
import com.googlecode.snoopyd.core.event.SnoopydStartedEvent;
import com.googlecode.snoopyd.core.event.SnoopydTerminatedEvent;

public interface KernelHandler {

    public void handle(KernelEvent event);

    public void handle(NetworkEnabledEvent event);

    public void handle(NetworkDisabledEvent event);

    public void handle(ChildSessionSendedEvent event);

    public void handle(ChildSessionRecivedEvent event);

    public void handle(DiscoverRecivedEvent event);

    public void handle(ParentNodeDeadedEvent event);

    public void handle(ChildNodeDeadedEvent event);

    public void handle(SnoopydStartedEvent event);

    public void handle(SnoopydTerminatedEvent event);

    public void handle(KernelStateChangedEvent event);

    public void handle(InvokationEvent event);

    public void handle(ScheduleTimeComeEvent event);

    public void handle(ExceptionEvent event);

    public void handle(ScheduleUpdatedEvent event);

    public void handle(ParentSessionRecivedEvent event);

    public void handle(ParentSessionSendedEvent event);

    public void handle(ResultRecievedEvent event);

    public void handle(ForceStartEvent event);

    public void handle(KernelReconfiguredEvent event);

}

```

<OfflineHandler.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.snoopyd.core.handler;  
  
import com.googlecode.snoopyd.core.Kernel;  
import com.googlecode.snoopyd.core.event.ChildNodeDeadedEvent;  
import com.googlecode.snoopyd.core.event.DiscoverRecivedEvent;  
import com.googlecode.snoopyd.core.event.KernelStateChangedEvent;  
import com.googlecode.snoopyd.core.event.NetworkDisabledEvent;  
import com.googlecode.snoopyd.core.event.NetworkEnabledEvent;  
import com.googlecode.snoopyd.core.event.ParentNodeDeadedEvent;  
import com.googlecode.snoopyd.core.event.ScheduleTimeComeEvent;  
import com.googlecode.snoopyd.core.state.OnlineState;  
  
public class OfflineHandler extends AbstractHandler implements  
    KernelHandler {  
  
    public OfflineHandler(Kernel kernel) {  
        super(kernel);  
    }  
  
    @Override  
    public void handle(NetworkEnabledEvent event) {  
  
        kernel.handle(new KernelStateChangedEvent(new OnlineState(kernel)));  
    }  
  
    @Override  
    public void handle(NetworkDisabledEvent event) {  
    }  
  
    @Override  
    public void handle(DiscoverRecivedEvent event) {  
    }  
  
    @Override  
    public void handle(ParentNodeDeadedEvent event) {  
    }  
  
    @Override  
    public void handle(ScheduleTimeComeEvent event) {
```

```

        }

    @Override
    public void handle(ChildNodeDeadedEvent event) {

    }
}

```

<OnlineHandler.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopyd.core.handler;

import java.util.HashMap;
import java.util.Map;

import com.googlecode.snoopyd.Defaults;
import com.googlecode.snoopyd.core.Kernel;
import com.googlecode.snoopyd.core.event.ChildNodeDeadedEvent;
import com.googlecode.snoopyd.core.event.ChildSessionSendedEvent;
import com.googlecode.snoopyd.core.event.DiscoverRecviedEvent;
import com.googlecode.snoopyd.core.event.KernelStateChangedEvent;
import com.googlecode.snoopyd.core.event.NetworkDisabledEvent;
import com.googlecode.snoopyd.core.event.NetworkEnabledEvent;
import com.googlecode.snoopyd.core.event.ParentNodeDeadedEvent;
import com.googlecode.snoopyd.core.event.ParentSessionReceivedEvent;
import com.googlecode.snoopyd.core.event.ParentSessionSendedEvent;
import com.googlecode.snoopyd.core.event.ScheduleTimeComeEvent;
import com.googlecode.snoopyd.core.event.ScheduleUpdatedEvent;
import com.googlecode.snoopyd.core.state.ActiveState;
import com.googlecode.snoopyd.core.state.OfflineState;
import com.googlecode.snoopyd.core.state.PassiveState;
import com.googlecode.snoopyd.driver.ISessionierPrx;
import com.googlecode.snoopyd.driver.ISessionierPrxHelper;
import com.googlecode.snoopyd.session.IKernelSessionPrx;
import com.googlecode.snoopyd.session.IKernelSessionPrxHelper;
import com.googlecode.snoopyd.session.KernelSession;
import com.googlecode.snoopyd.session.KernelSessionAdapter;
import com.googlecode.snoopyd.util.Identities;

public class OnlineHandler extends AbstractHandler implements KernelHandler {

    private Map<Ice.Identity, Integer> leaders;

    private long startTimeStamp;

```



```

+ targetContext.get("primary");

try {
    ISessionierPrx prx = ISessionierPrxHelper
        .checkedCast(kernel.communicator().stringToProxy(
            proxy));
    IKernelSessionPrx selfSession = IKernelSessionPrxHelper
        .uncheckedCast(kernel.primary()).addWithUUID(
            new KernelSessionAdapter(new
KernelSession(
            kernel))));

    IKernelSessionPrx leaderSession = prx.createKernelSession(
        kernel.identity(), selfSession);

    connected = true;
    kernel.handle(new ChildSessionSendedEvent(
        kernel.identity(), selfSession));
    kernel.handle(new
ParentSessionReceivedEvent(leaderIdentity,
            leaderSession));

    } catch (Ice.ConnectionLostException ex) {
        connected = false;
    } catch (Ice.ConnectFailedException ex) {
        connected = false;
    } catch (Ice.ConnectionTimeoutException ex) {
        connected = false;
    }
}
}

@Override
public void handle(ParentNodeDeadedEvent event) {

}

@Override
public void handle(ChildNodeDeadedEvent event) {

}

@Override
public void handle(ScheduleTimeComeEvent event) {

}

@Override
public void handle(ParentSessionReceivedEvent event) {
    super.handle(event);

    if (event.identity().equals(kernel.identity())) {
        kernel.handle(new KernelStateChangedEvent(new ActiveState(kernel)));
    } else {
        kernel.handle(new KernelStateChangedEvent(new PassiveState(kernel)));
    }
}

```

```

        kernel.handle(new ScheduleUpdatedEvent());
    }
}

<PassiveHadnler.java>

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopyd.core.handler;

import com.googlecode.snoopyd.coreKernel;
import com.googlecode.snoopyd.core.event.ChildNodeDeadedEvent;
import com.googlecode.snoopyd.core.event.DiscoverRecivedEvent;
import com.googlecode.snoopyd.core.event.KernelStateChangedEvent;
import com.googlecode.snoopyd.core.event.NetworkDisabledEvent;
import com.googlecode.snoopyd.core.event.NetworkEnabledEvent;
import com.googlecode.snoopyd.core.event.ParentNodeDeadedEvent;
import com.googlecode.snoopyd.core.event.ParentSessionRecivedEvent;
import com.googlecode.snoopyd.core.event.ParentSessionSendedEvent;
import com.googlecode.snoopyd.core.event.ScheduleTimeComeEvent;
import com.googlecode.snoopyd.core.state.ActiveState;
import com.googlecode.snoopyd.core.state.OfflineState;
import com.googlecode.snoopyd.core.state.OnlineState;
import com.googlecode.snoopyd.driver.ISessionierPrx;
import com.googlecode.snoopyd.driver.ISessionierPrxHelper;
import com.googlecode.snoopyd.session.IKernelSessionPrx;
import com.googlecode.snoopyd.session.IKernelSessionPrxHelper;
import com.googlecode.snoopyd.session.KernelSession;
import com.googlecode.snoopyd.session.KernelSessionAdapter;
import com.googlecode.snoopyd.util.Identities;

public class PassiveHandler extends AbstractHandler implements KernelHandler {

    public PassiveHandler(Kernel kernel) {
        super(kernel);
    }

    @Override
    public void handle(NetworkEnabledEvent event) {

    }

    @Override
    public void handle(NetworkDisabledEvent event) {

        kernel.childs().clear();
    }
}

```

```

        kernel.parents().clear();

        String proxy = Identities.toString(kernel.identity()) + ":" +
                      kernel.primaryEndpoints();

        ISessionierPrx prx = ISessionierPrxHelper.checkedCast(kernel
            .communicator().stringToProxy(proxy));

        IKernelSessionPrx selfSession = IKernelSessionPrxHelper
            .uncheckedCast(kernel.primary()).addWithUUID(
                new KernelSessionAdapter(new
        KernelSession(kernel))));

        IKernelSessionPrx remoteSession = prx.createKernelSession(
            kernel.identity(), selfSession);

        kernel.parents().put(kernel.identity(), remoteSession);

        kernel.handle(new KernelStateChangedEvent(new OfflineState(kernel)));
    }

    @Override
    public void handle(DiscoverReceivedEvent event) {
        kernel.cache().put(event.identity(), event.context());
    }

    @Override
    public void handle(ParentSessionReceivedEvent event) {
        super.handle(event);
    }

    @Override
    public void handle(ParentSessionSendedEvent event) {
        super.handle(event);
        kernel.handle(new KernelStateChangedEvent(new ActiveState(kernel)));
    }

    @Override
    public void handle(ParentNodeDeadedEvent event) {
        kernel.cache().clear();
        kernel.parents().remove(event.identity());
        kernel.handle(new KernelStateChangedEvent(new OnlineState(kernel)));
    }

    @Override
    public void handle(ChildNodeDeadedEvent event) {

    }

    @Override
    public void handle(ScheduleTimeComeEvent event) {

    }
}

```

<SuspenseHandler.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");

```

```

* You may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*      http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

```

```

package com.googlecode.snoopyd.core.handler;

import com.googlecode.snoopyd.core.Kernel;
import com.googlecode.snoopyd.core.event.ChildNodeDeadedEvent;
import com.googlecode.snoopyd.core.event.ChildSessionRecvEvent;
import com.googlecode.snoopyd.core.event.ChildSessionSendEvent;
import com.googlecode.snoopyd.core.event.DiscoverRecvEvent;
import com.googlecode.snoopyd.core.event.KernelStateChangedEvent;
import com.googlecode.snoopyd.core.event.NetworkDisabledEvent;
import com.googlecode.snoopyd.core.event.NetworkEnabledEvent;
import com.googlecode.snoopyd.core.event.ParentNodeDeadedEvent;
import com.googlecode.snoopyd.core.event.ScheduleTimeComeEvent;
import com.googlecode.snoopyd.core.state.OfflineState;
import com.googlecode.snoopyd.core.state.OnlineState;
import com.googlecode.snoopyd.driver.ISessionierPrx;
import com.googlecode.snoopyd.driver.ISessionierPrxHelper;
import com.googlecode.snoopyd.session.IKernelSessionPrx;
import com.googlecode.snoopyd.session.IKernelSessionPrxHelper;
import com.googlecode.snoopyd.session.KernelSession;
import com.googlecode.snoopyd.session.KernelSessionAdapter;
import com.googlecode.snoopyd.util.Identities;

public class SuspenseHandler extends AbstractHandler implements KernelHandler {

    public static final int TRY = 3;
    public static final int SLEEP = 10000;

    public SuspenseHandler(Kernel kernel) {
        super(kernel);
    }

    @Override
    public void handle(NetworkEnabledEvent event) {

        kernel.handle(new KernelStateChangedEvent(new OnlineState(kernel)));
    }

    @Override
    public void handle(NetworkDisabledEvent event) {

        String proxy = Identities.toString(kernel.identity()) + ":" +
                      kernel.primaryEndpoints();

        ISessionierPrx prx = ISessionierPrxHelper.checkedCast(kernel
            .communicator().stringToProxy(proxy));

        IKernelSessionPrx selfSession = IKernelSessionPrxHelper
            .uncheckedCast(kernel.primary()).addWithUUID(

```

```

new KernelSessionAdapter(new
KernelSession(kernel))));

IKernelSessionPrx remoteSession = prx.createKernelSession(
    kernel.identity(), selfSession);

kernel.parents().put(kernel.identity(), remoteSession);

kernel.handle(new KernelStateChangedEvent(new OfflineState(kernel)));
}

@Override
public void handle(ChildSessionSendedEvent event) {

}

@Override
public void handle(ChildSessionReceivedEvent event) {

    kernel.childs().put(event.identity(), event.session());
}

@Override
public void handle(DiscoverReceivedEvent event) {

}

@Override
public void handle(ParentNodeDeadedEvent event) {

}

@Override
public void handle(ChildNodeDeadedEvent event) {

}

@Override
public void handle(ScheduleTimeComeEvent event) {

}
}

```

<AbstractKernelFilter.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.

```

```

*/
package com.googlecode.snoopyd.core.filter;

import com.googlecode.snoopyd.core.Kernel;

public abstract class AbstractKernelFilter implements KernelFilter {

    protected Kernel kernel;

    public AbstractKernelFilter(Kernel kernel) {
        this.kernel = kernel;
    }

}

```

<KernelFilter.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopyd.core.filter;

import com.googlecode.snoopyd.core.event.KernelEvent;

public interface KernelFilter {

    public static enum FilterAction {
        ACCEPT, REJECT
    };

    public FilterAction accept(KernelEvent event);

}

```

<ToggleFilter.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and

```

```

 * limitations under the License.
 */
package com.googlecode.snoopyd.core.filter;

import com.googlecode.snoopyd.core.Kernel;
import com.googlecode.snoopyd.core.event.KernelEvent;
import com.googlecode.snoopyd.core.event.KernelStateChangedEvent;

public class ToogleFilter extends AbstractKernelFilter implements KernelFilter {

    public ToogleFilter(Kernel kernel) {
        super(kernel);
    }

    @Override
    public FilterAction accept(KernelEvent event) {

        FilterAction action = FilterAction.ACCEPT;

        if (event instanceof KernelStateChangedEvent && kernel.peek() instanceof
KernelStateChangedEvent) {
            action = FilterAction.REJECT;
        }

        return action;
    }
}

```

<ChildNodeDeadedEvent.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopyd.core.event;

import Ice.Identity;

public class ChildNodeDeadedEvent implements KernelEvent {

    private Ice.Identity identity;

    public ChildNodeDeadedEvent(Identity identity) {
        this.identity = identity;
    }

    public Ice.Identity identity() {
        return identity;
    }
}

```

```

@Override
public String name() {
    return this.getClass().getSimpleName();
}
}

<ChildSessionReceivedEvent.java>

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopyd.core.event;

import Ice.Identity;

import com.googlecode.snoopyd.session.ISessionPrx;

public class ChildSessionReceivedEvent implements KernelEvent {

    private Ice.Identity identity;
    private ISessionPrx session;

    public ChildSessionReceivedEvent(Identity identity, ISessionPrx session) {
        this.identity = identity;
        this.session = session;
    }

    @Override
    public String name() {
        return this.getClass().getSimpleName();
    }

    public ISessionPrx session() {
        return session;
    }

    public Ice.Identity identity() {
        return identity;
    }
}

```

<ChildSessionSendedEvent.java>

```

/**
 * Copyright 2011 Snoopy Project
 *

```

```

* Licensed under the Apache License, Version 2.0 (the "License");
* You may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*      http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

```

```

package com.googlecode.snoopyd.core.event;

import Ice.Identity;

import com.googlecode.snoopyd.session.ISessionPrx;

public class ChildSessionSendedEvent implements KernelEvent {

    private Identity identity;
    private ISessionPrx session;

    public ChildSessionSendedEvent(Identity identity, ISessionPrx session) {
        this.identity = identity;
        this.session = session;
    }

    @Override
    public String name() {
        return this.getClass().getSimpleName();
    }

    public ISessionPrx session() {
        return session;
    }

    public Ice.Identity identity() {
        return identity;
    }
}

```

<DiscoverReceivedEvent.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
*/

```

```

package com.googlecode.snoopyd.core.event;

import java.util.Map;

public class DiscoverReceivedEvent implements KernelEvent {

    private Map<String, String> context;
    private Ice.Identity identity;

    public DiscoverReceivedEvent(Ice.Identity identity, Map<String, String> context) {
        this.context = context;
        this.identity = identity;
    }

    @Override
    public String name() {
        return this.getClass().getSimpleName();
    }

    public Map<String, String> context() {
        return context;
    }

    public Ice.Identity identity() {
        return identity;
    }

}

```

<ExceptionEvent.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopyd.core.event;

public class ExceptionEvent implements KernelEvent {

    private Throwable exception;

    public ExceptionEvent(Throwable exception) {
        super();
        this.exception = exception;
    }

    public Throwable exception() {
        return exception;
    }

```

```

        @Override
        public String name() {
            return this.getClass().getSimpleName();
        }
    }

<ForceStartEvent.java>

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopyd.core.event;

public class ForceStartEvent implements KernelEvent {

    private String muid;
    private String[] params;

    public ForceStartEvent(String muid, String[] params) {
        this.muid = muid;
        this.params = params;
    }

    public String muid() {
        return muid;
    }

    public String[] params() {
        return params;
    }

    @Override
    public String name() {
        return this.getClass().getSimpleName();
    }
}

```

<InvocationEvent.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0

```

```

*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
package com.googlecode.snoopyd.core.event;

public abstract class InvokationEvent implements KernelEvent, Runnable {

    @Override
    public abstract void run();

    @Override
    public String name() {
        return InvokationEvent.class.getSimpleName();
    }

}

```

<KernelEvent.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
*/
package com.googlecode.snoopyd.core.event;

public interface KernelEvent {

    public String name();

}

```

<KernelReconfiguredEvent.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

```

```

* See the License for the specific language governing permissions and
* limitations under the License.
*/
package com.googlecode.snoopyd.core.event;

import java.util.Map;

public class KernelReconfiguredEvent implements KernelEvent {

    private Map<String, String> configuration;

    public KernelReconfiguredEvent(Map<String, String> configuration) {
        this.configuration = configuration;
    }

    public Map<String, String> configuration() {
        return configuration;
    }

    @Override
    public String name() {
        return this.getClass().getSimpleName();
    }
}

```

<KernelStateChangedEvent.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopyd.core.event;

import com.googlecode.snoopyd.core.state.KernelState;

public class KernelStateChangedEvent implements KernelEvent {

    private KernelState state;

    public KernelStateChangedEvent(KernelState state) {
        this.state = state;
    }

    @Override
    public String name() {
        return this.getClass().getSimpleName();
    }

    public KernelState state() {
        return state;
    }
}

```

```
    }  
}
```

<NetworkDisabledEvent.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *      http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.snoopyd.core.event;  
  
public class NetworkDisabledEvent implements KernelEvent {  
  
    @Override  
    public String name() {  
        return this.getClass().getSimpleName();  
    }  
}
```

<NetworkEnabledEvent.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *      http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.snoopyd.core.event;  
  
public class NetworkEnabledEvent implements KernelEvent {  
  
    @Override  
    public String name() {  
        return this.getClass().getSimpleName();  
    }  
}
```

<ParenNodeDeadEvent.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.snoopyd.core.event;  
  
import Ice.Identity;  
  
public class ParentNodeDeadedEvent implements KernelEvent {  
  
    private Ice.Identity identity;  
  
    public ParentNodeDeadedEvent(Identity identity) {  
        this.identity = identity;  
    }  
  
    public Ice.Identity identity() {  
        return identity;  
    }  
  
    @Override  
    public String name() {  
        return this.getClass().getSimpleName();  
    }  
}
```

<ParentSessionReceivedEvent.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.snoopyd.core.event;  
  
import Ice.Identity;
```

```

import com.googlecode.snoopyd.session.IKernelSessionPrx;

public class ParentSessionReceivedEvent implements KernelEvent {

    private Ice.Identity identity;
    private IKernelSessionPrx session;

    public ParentSessionReceivedEvent(Identity identity,
                                      IKernelSessionPrx session) {
        this.identity = identity;
        this.session = session;
    }

    @Override
    public String name() {
        return this.getClass().getSimpleName();
    }

    public Ice.Identity identity() {
        return identity;
    }

    public IKernelSessionPrx sesssion() {
        return session;
    }
}

```

<ParentSessionSendedEvent.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopyd.core.event;

import com.googlecode.snoopyd.session.IKernelSessionPrx;

public class ParentSessionSendedEvent implements KernelEvent {

    private Ice.Identity identity;
    private IKernelSessionPrx session;

    public ParentSessionSendedEvent(Ice.Identity identity, IKernelSessionPrx session) {
        this.identity = identity;
        this.session = session;
    }

    @Override

```

```

    public String name() {
        return this.getClass().getSimpleName();
    }

    public IKernelSessionPrx session() {
        return session;
    }

    public Ice.Identity identity() {
        return identity;
    }
}

```

<ResultReceivedEvent.java>

```

/*
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopyd.core.event;

import Ice.Identity;

public class ResultRecievedEvent implements KernelEvent {

    private Ice.Identity identity;
    private String muid;
    private String[] result;

    public ResultRecievedEvent(Identity identity, String muid, String[] result) {
        super();
        this.identity = identity;
        this.muid = muid;
        this.result = result;
    }

    public Ice.Identity identity() {
        return identity;
    }

    public String muid() {
        return muid;
    }

    public String[] result() {
        return result;
    }

    @Override
    public String name() {

```

```
        return this.getClass().getSimpleName();
    }
}
```

<ScheduleTimeComeEvent.java>

```
/*
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopyd.core.event;

public class ScheduleTimeComeEvent implements KernelEvent {

    private Ice.Identity identity;
    private String muid;
    private String[] params;

    public ScheduleTimeComeEvent(Ice.Identity identity, String muid, String[] params) {
        this.identity = identity;
        this.muid = muid;
        this.params = params;
    }

    @Override
    public String name() {
        return this.getClass().getSimpleName();
    }

    public Ice.Identity identity() {
        return identity;
    }

    public String muid() {
        return muid;
    }

    public String[] params() {
        return params;
    }
}
```

<ScheduleUpdatedEvent.java>

```
/*
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
```

```

/*
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopyd.core.event;

import Ice.Identity;

public class ScheduleUpdatedEvent implements KernelEvent {

    @Override
    public String name() {
        return this.getClass().getSimpleName();
    }
}

```

<SnoopyStartedEvent.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

```

```

package com.googlecode.snoopyd.core.event;

public class SnoopydStartedEvent implements KernelEvent {

    @Override
    public String name() {
        return this.getClass().getSimpleName();
    }
}

```

<SnoopyTerminatedEvent.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software

```

```

 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopyd.core.event;

public class SnoopydTerminatedEvent implements KernelEvent {

    @Override
    public String name() {
        return this.getClass().getSimpleName();
    }
}

```

<AbstractDriver.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopyd.driver;

import com.googlecode.snoopyd.core.Kernel;

public abstract class AbstractDriver implements Driver {

    protected Kernel kernel;
    protected String name;

    public AbstractDriver(String name, Kernel kernel) {
        this.name = name;
        this.kernel = kernel;
    }

    @Override
    public Kernel kernel() {
        return kernel;
    }

    @Override
    public String name() {
        return name;
    }
}

```

<Activable.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.snoopyd.driver;  
  
public interface Activable {  
  
    public void activate();  
    public void deactivate();  
}
```

<Aliver.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.snoopyd.driver;  
  
import java.util.Map;  
  
import org.apache.log4j.Logger;  
  
import com.googlecode.snoopyd.Defaults;  
import com.googlecode.snoopyd.core.Kernel;  
import com.googlecode.snoopyd.core.event.ChildNodeDeadedEvent;  
import com.googlecode.snoopyd.core.event.ParentNodeDeadedEvent;  
import com.googlecode.snoopyd.core.state.ActiveState;  
import com.googlecode.snoopyd.core.state.KernelListener;  
import com.googlecode.snoopyd.core.state.KernelState;  
import com.googlecode.snoopyd.core.state.PassiveState;  
import com.googlecode.snoopyd.session.IKernelSessionPrx;  
import com.googlecode.snoopyd.session.ISessionPrx;
```

```

import com.googlecode.snoopyd.util.Identities;

public class Aliver extends AbstractDriver implements Driver, Runnable,
    Startable, KernelListener {

    private static Logger logger = Logger.getLogger(Aliver.class);

    private boolean started;

    public Aliver(Kernel kernel) {
        super(Aliver.class.getSimpleName(), kernel);
        this.started = false;
    }

    @Override
    public void run() {

        try {

            for (; started;) {

                Map<Ice.Identity, ISessionPrx> parents = kernel.parents();
                for (Ice.Identity identity : parents.keySet()) {
                    IKernelSessionPrx parent = (IKernelSessionPrx) parents
                        .get(identity);

                    try {
                        parent.ice_ping();
                        logger.debug("parent node is alive: "
                            + Identities.toString(identity));
                    } catch (Exception ex) {
                        logger.debug("parent node is dead: "
                            + Identities.toString(identity));

                        kernel.handle(new ParentNodeDeadedEvent(identity));
                    }
                }

                Map<Ice.Identity, ISessionPrx> childs = kernel.childs();
                for (Ice.Identity identity : childs.keySet()) {
                    IKernelSessionPrx child = (IKernelSessionPrx) childs
                        .get(identity);

                    try {
                        child.ice_ping();
                        logger.debug("child node is alive: "
                            + Identities.toString(identity));
                    } catch (Exception ex) {
                        logger.debug("child node is dead: "
                            + Identities.toString(identity));

                        kernel.handle(new ChildNodeDeadedEvent(identity));
                    }
                }

                Thread.sleep(Defaults.ALIVE_INTERVAL);
            }

        } catch (InterruptedException ex) {
            logger.warn(ex.getMessage());
        }
    }
}

```

```

        synchronized (this) {
            notify();
        }
    }

@Override
public synchronized void start() {

    logger.debug("starting " + name);

    Thread self = new Thread(this, Defaults.ALIVER_THREAD_NAME);
    self.start();

    started = true;
}

@Override
public synchronized void stop() {

    logger.debug("stoping " + name);

    started = false;

    try {
        wait();
    } catch (InterruptedException e) {
        logger.warn(e.getMessage());
    }
}

@Override
public synchronized void restart() {

    logger.debug("restarting " + name);

    stop();
    start();
}

@Override
public boolean started() {
    return started;
}

@Override
public void stateChanged(KernelState currentState) {

    if (currentState instanceof ActiveState
            || currentState instanceof PassiveState) {
        if (!started) {
            start();
        }
    } else {
        if (started) {
            stop();
        }
    }
}
}

```

<Configurer.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
package com.googlecode.snoopyd.driver;  
  
import java.util.HashMap;  
import java.util.Map;  
import java.util.Properties;  
  
import org.apache.log4j.Logger;  
  
import com.googlecode.snoopyd.core.Kernel;  
import com.googlecode.snoopyd.core.event.KernelReconfiguredEvent;  
  
public class Configurer extends AbstractDriver implements Driver {  
    private static Logger logger = Logger.getLogger(Configurer.class);  
  
    public Configurer(Kernel kernel) {  
        super(Configurer.class.getSimpleName(), kernel);  
    }  
  
    public void reconfigure(Map<String, String> configuration) {  
        kernel.handle(new KernelReconfiguredEvent(configuration));  
    }  
  
    public Map<String, String> configuration() {  
        Properties prop = kernel.configuration();  
  
        Map<String, String> result = new HashMap<String, String>();  
        for (Object key: prop.keySet()) {  
            result.put(key.toString(),  
prop.getProperty(key.toString()).toString());  
        }  
  
        return result;  
    }  
}
```

<Controller.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.
```

```

* You may obtain a copy of the License at
*
*      http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
package com.googlecode.snoopyd.driver;

import org.apache.log4j.Logger;

import com.googlecode.snoopyd.core.Kernel;
import com.googlecode.snoopyd.core.event.SnoopydTerminatedEvent;

public class Controller extends AbstractDriver implements Driver {

    private static Logger logger = Logger.getLogger(Controller.class);

    public Controller(Kernel kernel) {
        super(Controller.class.getSimpleName(), kernel);
    }

    public void shutdown() {

        logger.debug("shutdown command received");

        kernel.handle(new SnoopydTerminatedEvent());
    }
}

```

<Discoverer.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
*/
package com.googlecode.snoopyd.driver;

import java.util.HashMap;
import java.util.Map;

import org.apache.log4j.Logger;

import com.googlecode.snoopyd.Defaults;
import com.googlecode.snoopyd.core.Kernel;

```

```

import com.googlecode.snoopyd.core.event.DiscoverReceivedEvent;
import com.googlecode.snoopyd.core.state.KernelListener;
import com.googlecode.snoopyd.core.state.KernelState;
import com.googlecode.snoopyd.core.state.OfflineState;
import com.googlecode.snoopyd.core.state.OnlineState;
import com.googlecode.snoopyd.core.state.SuspenseState;
import com.googlecode.snoopyd.util.Identities;

public class Discoverer extends AbstractDriver implements Driver, Runnable,
    Startable, KernelListener {

    private static Logger logger = Logger.getLogger(Discoverer.class);

    private boolean started;

    public Discoverer(Kernel kernel) {
        super(Discoverer.class.getSimpleName(), kernel);
        this.started = false;
    }

    public void discover(Ice.Identity identity, Map<String, String> context) {
        StringBuilder sb = new StringBuilder();
        for (String key: context.keySet()) {
            sb.append(key + "=" + context.get(key) + ":");
        }

        logger.debug("receive discover (" + sb.toString() + ") from " +
Identities.toString(identity));

        kernel.handle(new DiscoverReceivedEvent(identity, context));
    }

    @Override
    public synchronized void start() {

        logger.debug("starting " + name);

        Thread self = new Thread(this, Defaults.DISCOVERER_THREAD_NAME);
        self.start();

        started = true;
    }

    @Override
    public synchronized void stop() {

        logger.debug("stoping " + name);

        started = false;

        try {
            wait();
        } catch (InterruptedException e) {
            logger.warn(e.getMessage());
        }
    }

    @Override
    public synchronized void restart() {

        logger.debug("restarting " + name);
    }
}

```

```

        stop();
        start();
    }

@Override
public boolean started() {
    return started;
}

@Override
public void run() {

    com.googlecode.snoopyd.driver.IDiscovererPrx          dmulticast      =
com.googlecode.snoopyd.driver.IDiscovererPrxHelper
        .uncheckedCast(kernel.communicator().propertyToProxy(
            "Discoverer.Multicast"));
    dmulticast = com.googlecode.snoopyd.driver.IDiscovererPrxHelper
        .checkedCast(dmulticast.ice_datagram());

    com.googlecode.snoopycp.core.IDiscovererPrx          cpmulticast      =
com.googlecode.snoopycp.core.IDiscovererPrxHelper
        .uncheckedCast(kernel.communicator().propertyToProxy(
            "ControlPanel.Multicast"));
    cpmulticast = com.googlecode.snoopycp.core.IDiscovererPrxHelper
        .checkedCast(cpmulticast.ice_datagram());

try {

    for (; started;) {

        Map<String, String> context = new HashMap<String, String>();

        context.put("identity", Identities.toString(kernel.identity()));
        context.put("hostname", kernel.hostname());
        context.put("os", kernel.os());
        context.put("rate", String.valueOf(kernel.rate()));
        context.put("primary", kernel.primaryPublishedEndpoints());
        context.put("secondary", kernel.secondaryPublishedEndpoints());
        context.put("state", kernel.state().getClassName());

        StringBuilder sbchilds = new StringBuilder();
        for (Ice.Identity identity : kernel.childs().keySet()) {
            sbchilds.append(Identity.toString(identity));
            sbchilds.append(";");
        }
        context.put("childs", sbchilds.toString());

        StringBuilder sbparents = new StringBuilder();
        for (Ice.Identity identity : kernel.parents().keySet()) {
            sbparents.append(Identity.toString(identity));
            sbparents.append(";");
        }
        context.put("parents", sbparents.toString());

        try {
            dmulticast.discover(kernel.identity(), context);
            cpmulticast.discover(kernel.identity(), context);
        } catch (Exception ignored) {

```

```

        }

        Thread.sleep(Defaults.DISCOVER_INTERVAL);
    }

} catch (InterruptedException ex) {
    logger.warn(ex.getMessage());
}

synchronized (this) {
    notify();
}
}

@Override
public void stateChanged(KernelState currentState) {
    if (currentState instanceof OnlineState) {
        if (!started) {
            start();
        }
    } else if (currentState instanceof OfflineState
               || currentState instanceof SuspenseState) {
        if (started) {
            stop();
        }
    }
}
}

```

<Driver.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

```

```

package com.googlecode.snoopyd.driver;

import com.googlecode.snoopyd.core.Kernel;

public interface Driver {
    public Kernel kernel();
    public String name();
}

```

<Hoster.java>

```

/**
 * Copyright 2011 Snoopy Project
 *

```

```

* Licensed under the Apache License, Version 2.0 (the "License");
* You may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*      http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
package com.googlecode.snoopyd.driver;

import java.util.HashMap;
import java.util.Map;

import org.apache.log4j.Logger;
import org.hyperic.sigar.Sigar;
import org.hyperic.sigar.SigarException;

import com.googlecode.snoopyd.core.Kernel;

public class Hoster extends AbstractDriver implements Driver {

    private static Logger logger = Logger.getLogger(Hoster.class);

    private Sigar sigar;

    public Hoster(Kernel kernel) {
        super(Hoster.class.getSimpleName(), kernel);

        this.sigar = new Sigar();
    }

    public Map<String, String> context() {

        Map<String, String> result = new HashMap<String, String>();

        try {

            Map<String, String> net = sigar.getNetInfo().toMap();
            Map<String, String> mem = sigar.getMem().toMap();
            Map<String, String> cpu = sigar.getCpuInfoList()[0].toMap();

            result.putAll(net);
            result.putAll(cpu);
            result.putAll(mem);

        } catch (SigarException e) {
            logger.error(e.getMessage());
        }

        return result;
    }
}

```

<Invoker.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
package com.googlecode.snoopyd.driver;  
  
import java.util.LinkedList;  
import java.util.Map;  
import java.util.Queue;  
  
import org.apache.log4j.Logger;  
  
import com.googlecode.snoopyd.Defaults;  
import com.googlecode.snoopyd.core.Kernel;  
import com.googlecode.snoopyd.core.state.ActiveState;  
import com.googlecode.snoopyd.core.state.KernelListener;  
import com.googlecode.snoopyd.core.state.KernelState;  
import com.googlecode.snoopyd.core.state.PassiveState;  
  
/*  
 * TODO: remove it  
 */  
  
public class Invoker extends AbstractDriver implements Driver, Runnable,  
    Startable, KernelListener {  
  
    private static Logger logger = Logger.getLogger(Invoker.class);  
  
    private boolean started;  
  
    private Queue<Map<String, String>> pool;  
  
    public Invoker(Kernel kernel) {  
        super(Invoker.class.getSimpleName(), kernel);  
        this.started = false;  
        this.pool = new LinkedList<Map<String, String>>();  
    }  
  
    public synchronized void invoke(Map<String, String> invokation) {  
  
        pool.offer(invokation);  
        notify();  
    }  
  
    @Override  
    public synchronized void start() {  
        logger.debug("starting " + name);  
    }
```

```

        Thread self = new Thread(this, Defaults.INVOKER_THREAD_NAME);
        self.start();

    }

@Override
public synchronized void stop() {
    logger.debug("stoping " + name);

    started = false;
    notify();

    try {
        wait();
    } catch (InterruptedException e) {
        logger.warn(e.getMessage());
    }
}

@Override
public synchronized void restart() {
    logger.debug("restarting " + name);

    stop();
    start();
}

@Override
public boolean started() {
    return started;
}

@Override
public void run() {

    for (;started;) {

        for (; !pool.isEmpty() && started;) {

            /*
             * module = <MODULE_UUID>
             * params = <PARAM1>, <PARAM2>
             */
            Map<String, String> invokation = pool.poll();

            // TODO: Hard-Code HERE! Bitch!
        }

        if (started) {
            synchronized (this) {
                try {
                    wait();
                } catch (InterruptedException ignored) {
                }
            }
        }
    }
}

```

```

        synchronized (this) {
            notify();
        }
    }

@Override
public void stateChanged(KernelState currentState) {
    if (currentState instanceof ActiveState
        || currentState instanceof PassiveState) {
        if (!started) {
            start();
        }
    } else {
        if (started) {
            stop();
        }
    }
}
}

```

<Moduler.java>

```

/*
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopyd.driver;

import java.util.Map;

import org.apache.log4j.Logger;

import com.googlecode.snoopyd.core.Kernel;
import com.googlecode.snoopyd.core.event.ForceStartEvent;
import com.googlecode.snoopymm.ModuleNotFoundException;

public class Moduler extends AbstractDriver implements Driver {

    private static Logger logger = Logger.getLogger(Moduler.class);

    public Moduler(Kernel kernel) {
        super(Moduler.class.getSimpleName(), kernel);
    }

    public Map<String, String> fetch() {
        return kernel.moduleManager().fetch();
    }

    public String[] launch(String muid, String[] params) throws
com.googlecode.snoopyd.driver.ModuleNotFoundException {

```

```

        try {
            return kernel.moduleManager().launch(muid, params);
        } catch (ModuleNotFoundException ex) {
            throw new com.googlecode.snoopyd.driver.ModuleNotFoundException();
        }
    }

    public void deploy(String muid, String code) {
        kernel.moduleManager().deploy(muid, code);
    }

    public void undeploy(String muid) throws ModuleNotFoundException {
        kernel.moduleManager().undeploy(muid);
    }

    public void force(String muid, String[] params) {
        kernel.handle(new ForceStartEvent(muid, params));
    }
}

```

<Networker.java>

```

/*
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopyd.driver;

import java.net.NetworkInterface;
import java.net.SocketException;
import java.util.Enumeration;

import org.apache.log4j.Logger;

import com.googlecode.snoopyd.Defaults;
import com.googlecode.snoopyd.core.Kernel;
import com.googlecode.snoopyd.core.event.NetworkDisabledEvent;
import com.googlecode.snoopyd.core.event.NetworkEnabledEvent;

public class Networker extends AbstractDriver implements Driver, Activable,
    Runnable {

    private static Logger logger = Logger.getLogger(Networker.class);

    public static final int NETWORK_UNDEFINED = -1;
    public static final int NETWORK_ENABLED = 0;
    public static final int NETWORK_DISABLED = 1;

    private int networkState;

```

```

private boolean started;

public Networker(Kernel kernel) {
    super(Networker.class.getSimpleName(), kernel);

    this.networkState = NETWORK_UNDEFINED;
    this.started = false;
}

@Override
public void run() {

    for (; started;) {

        try {

            boolean checker = false;

            Enumeration<NetworkInterface> interfaces = NetworkInterface
                .getNetworkInterfaces();

            while (interfaces.hasMoreElements()) {
                NetworkInterface nic = interfaces.nextElement();
                try {

                    logger.debug("checking " + nic.getDisplayName()
                        + " interface :: " + nic.isUp());

                    checker = checker || (nic.isUp() &&
!nic.isLoopback());

                } catch (SocketException ignored) {
                }
            }

            if (networkState == NETWORK_UNDEFINED) {
                if (checker) {

                    logger.debug("network is enabled");

                    networkState = NETWORK_ENABLED;

                    kernel.handle(new NetworkEnabledEvent());
                } else {

                    logger.debug("network is disabled");

                    networkState = NETWORK_DISABLED;

                    kernel.handle(new NetworkDisabledEvent());
                }
            } else if (networkState == NETWORK_DISABLED) {

                if (checker) {

                    logger.debug("network is enabled");

```

```

        networkState = NETWORK_ENABLED;

        kernel.handle(new NetworkEnabledEvent());

    } else {

        logger.debug("network still disabled");
    }

} else if (networkState == NETWORK_ENABLED) {

    if (!checker) {

        logger.debug("network is disabled");

        networkState = NETWORK_DISABLED;

        kernel.handle(new NetworkDisabledEvent());
    } else {

        logger.debug("network still enabled");
    }
}

} catch (SocketException ex) {

    if (networkState == NETWORK_UNDEFINED
        || networkState == NETWORK_ENABLED) {

        logger.debug("network is disabled");

        networkState = NETWORK_DISABLED;

        kernel.handle(new NetworkDisabledEvent());
    }
}

try {
    Thread.sleep(Defaults.NETWORKER_INTERVAL);
} catch (InterruptedException ignored) {
}
}

synchronized (this) {
    notify();
}
}

@Override
public synchronized void activate() {

    started = true;

    Thread self = new Thread(this);
    self.start();
}

@Override
public synchronized void deactivate() {

```

```

        started = false;

        try {
            wait();
        } catch (InterruptedException e) {
            logger.warn(e.getMessage());
        }
    }
}

```

<Resulter.java>

```

/*
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopyd.driver;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Arrays;
import java.util.LinkedList;
import java.util.Queue;

import org.apache.log4j.Logger;

import com.googlecode.snoopyd.Defaults;
import com.googlecode.snoopyd.core.Kernel;
import com.googlecode.snoopyd.core.state.ActiveState;
import com.googlecode.snoopyd.core.state.KernelListener;
import com.googlecode.snoopyd.core.stateKernelState;

public class Resulter extends AbstractDriver implements Driver, Startable,
    Runnable, KernelListener {

    private static Logger logger = Logger.getLogger(Resulter.class);

    public static class Result {

        private String hostname;
        private String osname;
        private String module;
        private String[] result;

        public Result(String hostname, String osname, String module, String[] result)
{
            this.hostname = hostname;
            this.osname = osname;

```

```

        this.module = module;
        this.result = result;
    }

    public String hostname() {
        return hostname;
    }

    public String osname() {
        return osname;
    }

    public String module() {
        return module;
    }

    public String[] result() {
        return result;
    }
}

private Connection connection;

private Queue<Result> pool;

private boolean started;

public Resulter(Kernel kernel) {
    super(Resulter.class.getSimpleName(), kernel);

    this.pool = new LinkedList<Result>();
    this.started = false;
}

public synchronized void store(String hostname, String osname, String module,
    String[] result) {

    logger.debug("storing result " + Arrays.toString(result));

    pool.offer(new Result(hostname, osname, module, result));

    if (pool.size() > Defaults.RESULTER_THRESHOLD) {
        notify();
    }
}

public synchronized void start() {
    logger.debug("starting " + name);

    try {

        String connectionurl
kernel.configuration().getProperty("connectionstring");

        logger.debug("use connection url: " + connectionurl);

        Class.forName("com.mysql.jdbc.Driver");
        connection = DriverManager.getConnection(connectionurl);

    } catch (ClassNotFoundException ex) {
        logger.error(ex.getMessage());
    }
}

```

```

        } catch (SQLException ex) {
            logger.error(ex.getMessage());
        }

        Thread self = new Thread(this, Defaults.RESULTER_THREAD_NAME);
        self.start();

        started = true;
    }

@Override
public synchronized void stop() {
    logger.debug("stoping " + name);

    try {
        connection.close();
    } catch (SQLException ex) {

    } catch (NullPointerException ex) {

    }

    started = false;
    notify();

    try {
        wait();
    } catch (InterruptedException e) {
        logger.warn(e.getMessage());
    }
}

@Override
public synchronized boolean started() {
    return started;
}

@Override
public synchronized void restart() {
    logger.debug("restarting " + name);

    stop();
    start();
}

@Override
public void run() {

    for (; started;) {

        synchronized (this) {
            try {
                wait();
            } catch (InterruptedException ignored) {
            }
        }

        for (; !pool.isEmpty() && started;) {

            Result result = pool.poll();

```

```

        try {
            StringBuilder sb = new StringBuilder();
            for (String res: result.result()) {
                sb.append(res);
                sb.append(";");
            }

            Statement stmt = connection.createStatement();
            stmt.executeUpdate(String.format("Call storeResult(\"%s\",
 \"%s\",    \"%s\",    \"%s\");",    result.osname(),    result.hostname(),    result.module(),
sb.toString()));
            stmt.close();

        } catch (SQLException ex) {
            logger.debug(ex.getMessage());
        }
    }

    synchronized (this) {
        notify();
    }
}

@Override
public void stateChanged(KernelState currentState) {
    if (currentState instanceof ActiveState) {
        if (!started) {
            start();
        }
    } else {
        if (started) {
            stop();
        }
    }
}
}

```

<Scheduler.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopyd.driver;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;

```

```

import java.io.IOException;
import java.io.OutputStreamWriter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Timer;
import java.util.TimerTask;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.apache.log4j.Logger;
import org.w3c.dom.Document;
import org.w3c.dom.DocumentType;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import com.googlecode.snoopyd.Defaults;
import com.googlecode.snoopyd.core.Kernel;
import com.googlecode.snoopyd.core.event.ScheduleTimeComeEvent;
import com.googlecode.snoopyd.core.event.ScheduleUpdatedEvent;
import com.googlecode.snoopyd.core.state.ActiveState;
import com.googlecode.snoopyd.core.state.KernelListener;
import com.googlecode.snoopyd.core.state.KernelState;
import com.googlecode.snoopyd.util.Identities;

public class Scheduler extends AbstractDriver implements Driver, Startable,
    Activable, KernelListener {

    public static enum ScheduleState {
        ON, OFF
    }

    public static class Schedule {

        private HashMap<String, List<Long>> timetable;
        private Map<String, ScheduleState> states;
        private Map<String, List<String>> params;

        public Schedule() {

            this.timetable = new HashMap<String, List<Long>>();
            this.states = new HashMap<String, Scheduler.ScheduleState>();
            this.params = new HashMap<String, List<String>>();
        }

        public HashMap<String, List<Long>> timetable() {
            return timetable;
        }
    }
}

```

```

        public Map<String, ScheduleState> statetable() {
            return states;
        }

        public Map<String, List<String>> paramstable() {
            return params;
        }
    }

    private static Logger logger = Logger.getLogger(Scheduler.class);

    private Schedule self;
    private HashMap<Ice.Identity, Schedule> childs;

    private Map<String, Timer> timers;

    private boolean started;

    public Scheduler(Kernel kernel) {
        super(Scheduler.class.getSimpleName(), kernel);

        this.self = new Schedule();
        this.childs = new HashMap<Ice.Identity, Schedule>();

        this.timers = new HashMap<String, Timer>();

        loadScheduleConfig();
    }

    public void synchronize(Ice.Identity identity, ISchedulerPrx remoteScheduler) {
        logger.debug("synchronize shceduler with "
                + Identities.toString(identity));

        Schedule childSchedule = new Schedule();

        Map<String, String> remoteTimetable = remoteScheduler.timetable();
        Map<String, String> remoteStatetable = remoteScheduler.statetable();
        Map<String, String> remoteParamtable = remoteScheduler.paramtable();

        for (String muid : remoteTimetable.keySet()) {
            List<Long> times = new ArrayList<Long>();
            for (String time : remoteTimetable.get(muid).split(";")) {
                times.add(Long.parseLong(time));
            }
            childSchedule.timetable().put(muid, times);
        }

        for (String muid : remoteStatetable.keySet()) {
            if (remoteStatetable.get(muid).equals("ON")) {
                childSchedule.statetable().put(muid, ScheduleState.ON);
            } else {
                childSchedule.statetable().put(muid, ScheduleState.OFF);
            }
        }

        for (String muid : remoteParamtable.keySet()) {
            List<String> params = new ArrayList<String>();
            for (String param : remoteParamtable.get(muid).split(";")) {
                params.add(param);
            }
        }
    }
}

```

```

        childSchedule.paramstable().put(muid, params);
    }

    childS.put(identity, childSchedule);

    update();
}

public void schedule(String muid, long[] delays, String[] params) {

    if (self.timetable().containsKey(muid)) {
        for (long delay: delays) {
            self.timetable().get(muid).add(delay);
        }
    } else {
        self.timetable().put(muid, new ArrayList<Long>());
        for (long delay: delays) {
            self.timetable().get(muid).add(delay);
        }
    }

    self.paramstable().put(muid, Arrays.asList(params));

    self.statetable().put(muid, ScheduleState.ON);
    kernel.handle(new ScheduleUpdatedEvent());
}

public void unschedule(String muid) throws ModuleNotFoundException {
    kernel.handle(new ScheduleUpdatedEvent());
}

public void force(Ice.Identity identity, String muid, String[] params) {
    kernel.handle(new ScheduleTimeComeEvent(identity, muid, params));
}

public Map<String, String> timetable() {

    Map<String, String> result = new HashMap<String, String>();

    for (String muid : self.timetable().keySet()) {

        StringBuilder sb = new StringBuilder();
        for (Long time : self.timetable().get(muid)) {
            sb.append(time);
            sb.append(";");
        }

        result.put(muid, sb.toString());
    }

    return result;
}

public Map<String, String> statetable() {

    Map<String, String> result = new HashMap<String, String>();

    for (String muid : self.statetable().keySet()) {
        result.put(muid, self.statetable().get(muid).toString());
    }
}

```

```

        return result;
    }

    public Map<String, String> paramtable() {
        Map<String, String> result = new HashMap<String, String>();
        for (String muid : self.paramstable().keySet()) {
            StringBuilder sb = new StringBuilder();
            for (String param : self.paramstable().get(muid)) {
                sb.append(param);
                sb.append(";");
            }
            result.put(muid, sb.toString());
        }
        return result;
    }

    public void toogle(String muid) {
        ScheduleState state = self.statetable().get(muid);

        if (state == ScheduleState.OFF) {
            self.statetable().put(muid, ScheduleState.ON);
        } else {
            self.statetable().put(muid, ScheduleState.OFF);
        }

        kernel.handle(new ScheduleUpdatedEvent());
    }

    public void cancel(Ice.Identity identity) {
        logger.debug("cancel scheduling for " + Identities.toString(identity));
        childs.remove(identity);
        update();
    }

    @Override
    public synchronized void start() {
        logger.debug("starting " + name);
        started = true;
    }

    @Override
    public synchronized void stop() {
        logger.debug("stoping " + name);

        for (String muid : timers.keySet()) {
            timers.get(muid).cancel();
        }
    }

    @Override
    public synchronized boolean started() {

```

```

        return started;
    }

    @Override
    public synchronized void restart() {

        logger.debug("restarting " + name);

        stop();
        start();
    }

    @Override
    public void stateChanged(KernelState currentState) {
        if (currentState instanceof ActiveState) {
            if (!started) {
                start();
            }
        } else {
            if (started) {
                stop();
            }
        }
    }

    @Override
    public void activate() {
        loadScheduleConfig();
    }

    @Override
    public void deactivate() {
        saveScheduleConfig();
    }

    private void update() {

        logger.debug("updating scheduler");

        synchronized (timers) {

            for (String muid : timers.keySet()) {
                timers.get(muid).cancel();
            }

            timers.clear();

            for (Ice.Identity identity : childs.keySet()) {
                logger.debug("set scheduler for node " +
Identities.toString(identity));

                Schedule childSchedule = childSchedule.timetable().keySet());
                final Ice.Identity fidentity = identity;

                for (String muid : childSchedule.timetable().keySet()) {
                    logger.debug("set schedule for module " + muid);
                }
            }
        }
    }
}

```

```

        if          (childSchedule.statetable().get(muid)      ==
ScheduleState.ON) {
            final String fmuid = muid;
            final String[] fprms = childSchedule
                .paramstable()
                .get(muid)
                .toArray(
                    new
String[childSchedule.paramstable()
    .get(muid).size()]);
            Timer timer = new Timer(Defaults.TIMER_THREAD_NAME
                + "-" + muid);
            for          (Long           delay      :
childSchedule.timetable().get(muid)) {
                timer.schedule(new TimerTask() {
                    @Override
                    public void run() {
                        kernel.handle(new
ScheduleTimeComeEvent(
                            fidentity,     fmuid,
fprms));
                    }
                }, 0, delay.longValue());
            }
            timers.put(muid, timer);
        }
    }
}

private void loadScheduleConfig() {
    try {
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        DocumentBuilder db = dbf.newDocumentBuilder();
        Document dom = db.parse(new File(kernel.properties().getProperty(
            "Snoopy.SchedulerConfiguration")));
        Element root = dom.getDocumentElement();

        NodeList modules = root.getElementsByTagName("module");
        for (int i = 0; i < modules.getLength(); i++) {
            Element module = (Element) modules.item(i);
            String muid = module.getAttribute("muid");
            String state = module.getAttribute("state");

            List<Long> delaysList = new ArrayList<Long>();
            NodeList delays = module.getElementsByTagName("delay");
            for (int j = 0; j < delays.getLength(); j++) {
                Element delay = (Element) delays.item(j);
                String value = delay.getAttribute("value");
                delaysList.add(Long.parseLong(value));
            }

            List<String> paramsList = new ArrayList<String>();
            NodeList prms = module.getElementsByTagName("param");
            for (int j = 0; j < prms.getLength(); j++) {
}

```

```

        Element param = (Element) prms.item(j);
        String value = param.getAttribute("value");
        paramsList.add(value);
    }

    self.timetable().put(muid, delaysList);
    self.paramstable().put(muid, paramsList);

    if (state.equals("ON")) {
        self.statetable().put(muid, ScheduleState.ON);
    } else {
        self.statetable().put(muid, ScheduleState.OFF);
    }
}

} catch (ParserConfigurationException ex) {

} catch (IOException ex) {

} catch (SAXException ex) {

}
}

private void saveScheduleConfig() {

try {

DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
DocumentBuilder db = dbf.newDocumentBuilder();
Document doc = db.newDocument();

Element root = doc.createElement("modules");
doc.appendChild(root);

for (String muid: self.timetable().keySet()) {
    Element module = doc.createElement("module");
    module.setAttribute("muid", muid);
    if (self.statetable().get(muid) == ScheduleState.ON) {
        module.setAttribute("state", "ON");
    } else {
        module.setAttribute("state", "OFF");
    }

    Element schedule = doc.createElement("schedule");
    for (Long time: self.timetable().get(muid)) {
        Element delay = doc.createElement("delay");
        delay.setAttribute("value", time.toString());
        schedule.appendChild(delay);
    }

    module.appendChild(schedule);

    Element params = doc.createElement("params");
    for (String value: self.paramstable().get(muid)) {
        Element param = doc.createElement("param");
        param.setAttribute("value", value);
        params.appendChild(param);
    }

    module.appendChild(params);
}
}
}

```

<Sessioner.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *      http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */
```

```
package com.googlecode.snoopyd.driver;  
  
import org.apache.log4j.Logger;  
  
import com.googlecode.snoopyd.core.Kernel;
```

```

import com.googlecode.snoopyd.core.event.ChildSessionReceivedEvent;
import com.googlecode.snoopyd.core.event.ParentSessionSendedEvent;
import com.googlecode.snoopyd.core.state.KernelListener;
import com.googlecode.snoopyd.core.state.KernelState;
import com.googlecode.snoopyd.session.IKernelSessionPrx;
import com.googlecode.snoopyd.session.IKernelSessionPrxHelper;
import com.googlecode.snoopyd.session.IUserSessionPrx;
import com.googlecode.snoopyd.session.IUserSessionPrxHelper;
import com.googlecode.snoopyd.session.KernelSession;
import com.googlecode.snoopyd.session.KernelSessionAdapter;
import com.googlecode.snoopyd.session.UserSession;
import com.googlecode.snoopyd.session.UserSessionAdapter;

public class Sessionier extends AbstractDriver implements Driver,
    KernelListener {

    private static Logger logger = Logger.getLogger(Sessionier.class);

    public Sessionier(Kernel kernel) {
        super(Sessionier.class.getSimpleName(), kernel);
    }

    public IKernelSessionPrx createKernelSession(Ice.Identity identity,
                                                IKernelSessionPrx selfSession) {

        kernel.handle(new ChildSessionReceivedEvent(identity, selfSession));

        IKernelSessionPrx remoteSession = IKernelSessionPrxHelper
            .uncheckedCast(kernel.primary()).addWithUUID(
                new                                         KernelSessionAdapter(new
KernelSession(kernel))));

        kernel.handle(new ParentSessionSendedEvent(kernel.identity(),
remoteSession));

        return remoteSession;
    }

    public IUserSessionPrx createUserSession(Ice.Identity identity,
                                            IUserSessionPrx selfSession) {

        //kernel.handle(new ChildSessionReceivedEvent(identity, selfSession));

        IUserSessionPrx remoteSession = IUserSessionPrxHelper
            .uncheckedCast(kernel.primary()).addWithUUID(
                new UserSessionAdapter(new UserSession(kernel))));

        //kernel.handle(new ChildSessionSendedEvent(identity, remoteSession));

        return remoteSession;
    }

    @Override
    public void stateChanged(KernelState currentState) {

    }
}

```

<Startable.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.snoopyd.driver;  
  
public interface Startable {  
  
    public void start();  
    public void stop();  
  
    public boolean started();  
  
    public void restart();  
}
```

<KernelSession.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.snoopyd.session;  
  
import com.googlecode.snoopyd.adapter.ModulerAdapter;  
import com.googlecode.snoopyd.adapter.SchedulerAdapter;  
import com.googlecode.snoopyd.core.Kernel;  
import com.googlecode.snoopyd.driver.IModulerPrx;  
import com.googlecode.snoopyd.driver.IModulerPrxHelper;  
import com.googlecode.snoopyd.driver.ISchedulerPrx;  
import com.googlecode.snoopyd.driver.ISchedulerPrxHelper;  
import com.googlecode.snoopyd.driver.Moduler;  
import com.googlecode.snoopyd.driver.Scheduler;  
  
public class KernelSession {
```

```

private Kernel kernel;

public KernelSession(Kernel kernel) {
    this.kernel = kernel;
}

public ISchedulerPrx scheduler() {

    ISchedulerPrx remoteScheduler = ISchedulerPrxHelper
        .uncheckedCast(kernel.primary()).addWithUUID(
            new SchedulerAdapter((Scheduler) kernel
                .driver(Scheduler.class))));

    return remoteScheduler;
}

public IModulerPrx moduler() {

    IModulerPrx remoteModuler = IModulerPrxHelper.uncheckedCast(kernel
        .primary()).addWithUUID(
            new ModulerAdapter((Moduler) kernel
                .driver(Moduler.class))));

    return remoteModuler;
}
}

```

<KernelSessionAdapter.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopyd.session;

import Ice.Current;

import com.googlecode.snoopyd.driver.IModulerPrx;
import com.googlecode.snoopyd.driver.ISchedulerPrx;

public class KernelSessionAdapter extends _IKernelSessionDisp {

    private KernelSession kernelSession;

    public KernelSessionAdapter(KernelSession kernelSession) {
        this.kernelSession = kernelSession;
    }
}

```

```

@Override
public void destroy(Current __current) {
}

@Override
public void refresh(Current __current) {
}

@Override
public IModulerPrx moduler(Current __current) {
    return kernelSession.moduler();
}

@Override
public ISchedulerPrx scheduler(Current __current) {
    return kernelSession.scheduler();
}

}

```

<UserSession.java>

```


/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */


package com.googlecode.snoopyd.session;

import com.googlecode.snoopyd.adapter.ConfigurerAdapter;
import com.googlecode.snoopyd.adapter.ControllerAdapter;
import com.googlecode.snoopyd.adapter.HostAdapter;
import com.googlecode.snoopyd.adapter.ModulerAdapter;
import com.googlecode.snoopyd.adapter.SchedulerAdapter;
import com.googlecode.snoopyd.coreKernel;
import com.googlecode.snoopyd.driver.Configurer;
import com.googlecode.snoopyd.driver.Controller;
import com.googlecode.snoopyd.driver.Host;
import com.googlecode.snoopyd.driver.IConfigurerPrx;
import com.googlecode.snoopyd.driver.IConfigurerPrxHelper;
import com.googlecode.snoopyd.driver.IControllerPrx;
import com.googlecode.snoopyd.driver.IControllerPrxHelper;
import com.googlecode.snoopyd.driver.IHosterPrx;
import com.googlecode.snoopyd.driver.IHosterPrxHelper;
import com.googlecode.snoopyd.driver.IModulerPrx;
import com.googlecode.snoopyd.driver.IModulerPrxHelper;
import com.googlecode.snoopyd.driver.ISchedulerPrx;

```

```

import com.googlecode.snoopyd.driver.ISchedulerPrxHelper;
import com.googlecode.snoopyd.driver.Moduler;
import com.googlecode.snoopyd.driver.Scheduler;

public class UserSession {

    private Kernel kernel;

    public UserSession(Kernel kernel) {
        this.kernel = kernel;
    }

    public IHosterPrx hoster() {

        IHosterPrx remoteHoster = IHosterPrxHelper
            .uncheckedCast(kernel.primary())
            .addWithUUID(
                new HosterAdapter((Hoster) kernel
                    .driver(Hoster.class))));

        return remoteHoster;
    }

    public IControllerPrx controller() {

        IControllerPrx remoteController = IControllerPrxHelper
            .uncheckedCast(kernel.primary()).addWithUUID(
                new ControllerAdapter((Controller) kernel
                    .driver(Controller.class))));

        return remoteController;
    }

    public IModulerPrx moduler() {

        IModulerPrx remoteModuler = IModulerPrxHelper.uncheckedCast(kernel
            .primary()).addWithUUID(
                new ModulerAdapter((Moduler) kernel
                    .driver(Moduler.class))));

        return remoteModuler;
    }

    public IConfigurerPrx configurer() {

        IConfigurerPrx remoteConfigurer = IConfigurerPrxHelper
            .uncheckedCast(kernel.primary()).addWithUUID(
                new ConfigurerAdapter((Configurer) kernel
                    .driver(Configurer.class))));

        return remoteConfigurer;
    }

    public ISchedulerPrx scheduler() {

        ISchedulerPrx remoteScheduler = ISchedulerPrxHelper
            .uncheckedCast(kernel.primary()).addWithUUID(
                new SchedulerAdapter((Scheduler) kernel
                    .driver(Scheduler.class))));
    }
}

```

```

        return remoteScheduler;
    }

}

<UserSessionAdapter.java>

/***
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopyd.session;

import com.googlecode.snoopyd.driver.IConfigurerPrx;
import com.googlecode.snoopyd.driver.IControllerPrx;
import com.googlecode.snoopyd.driver.IHosterPrx;
import com.googlecode.snoopyd.driver.IModulerPrx;
import com.googlecode.snoopyd.driver.ISchedulerPrx;

import Ice.Current;

public class UserSessionAdapter extends _IUserSessionDisp {

    private UserSession userSession;

    public UserSessionAdapter(UserSession userSession) {
        this.userSession = userSession;
    }

    @Override
    public void destroy(Current __current) {

    }

    @Override
    public void refresh(Current __current) {

    }

    @Override
    public IHosterPrx hoster(Current __current) {
        return userSession.hoster();
    }

    @Override
    public IControllerPrx controller(Current __current) {
        return userSession.controller();
    }
}

```

```

@Override
public IModulerPrx moduler(Current __current) {
    return userSession.moduler();
}

@Override
public IConfigurerPrx configurer(Current __current) {
    return userSession.configurer();
}

@Override
public ISchedulerPrx scheduler(Current __current) {
    return userSession.scheduler();
}
}

```

<AbstractModule.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopyd.module;

import java.util.UUID;

public abstract class AbstractModule implements Module {

    protected UUID uudi;
    protected String name;

    public AbstractModule(UUID uudi, String name) {
        this.uudi = uudi;
        this.name = name;
    }

    public UUID uudi() {
        return uudi;
    }

    public String name() {
        return name;
    }
}

```

<Module.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *      http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
package com.googlecode.snoopyd.module;  
  
public interface Module {  
  
    public Object invoke(Object arg);  
  
}
```

<PythonModule.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *      http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
package com.googlecode.snoopyd.module;  
  
import java.util.UUID;  
  
public class PythonModule extends AbstractModule implements Module {  
  
    public PythonModule(UUID uudi, String name) {  
        super(uudi, name);  
    }  
  
    @Override  
    public Object invoke(Object arg) {  
        // TODO Auto-generated method stub  
        return null;  
    }  
  
}
```

<Adapter.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.snoopyd.adapter;  
  
import com.googlecode.snoopyd.driver.Driver;  
  
public interface Adapter {  
  
    public Driver driver();  
    public String name();  
    public Ice.Identity identity();  
  
}
```

<ConfigurerAdapter.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.snoopyd.adapter;  
  
import java.util.Map;  
  
import Ice.Current;  
  
import com.googlecode.snoopyd.driver.Configurer;  
import com.googlecode.snoopyd.driver._IConfigurerDisp;  
  
public class ConfigurerAdapter extends _IConfigurerDisp {  
  
    private Configurer configurer;  
  
    public ConfigurerAdapter(Configurer configurer) {
```

```

        this.configurer = configurer;
    }

    @Override
    public void reconfigure(Map<String, String> configuration, Current __current) {
        configurer.reconfigure(configuration);
    }

    @Override
    public Map<String, String> configuration(Current __current) {
        return configurer.configuration();
    }
}

```

<ControllerAdapter.java>

```

/*
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopyd.adapter;

import Ice.Current;

import com.googlecode.snoopyd.driver.Controller;
import com.googlecode.snoopyd.driver._IControllerDisp;

public class ControllerAdapter extends _IControllerDisp {

    private Controller controller;

    public ControllerAdapter(Controller controller) {
        this.controller = controller;
    }

    @Override
    public void shutdown(Current __current) {
        controller.shutdown();
    }
}

```

<DiscovererAdapter.java>

```

/*
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *

```

```

*      http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
package com.googlecode.snoopyd.adapter;

import org.apache.log4j.Logger;

import Ice.Current;
import Ice.Identity;

import com.googlecode.snoopyd.driver.Discoverer;
import com.googlecode.snoopyd.driver.Driver;
import com.googlecode.snoopyd.driver._IDiscovererDisp;

public class DiscovererAdapter extends _IDiscovererDisp implements Adapter {

    private static Logger logger = Logger.getLogger(DiscovererAdapter.class);

    private String name;
    private Ice.Identity identity;

    private Discoverer discoverer;

    public DiscovererAdapter(Ice.Identity identity, Discoverer discoverer) {
        this.discoverer = discoverer;
        this.name = DiscovererAdapter.class.getSimpleName();
        this.identity = identity;
    }

    @Override
    public void discover(Ice.Identity identity, Current __current) {

        discoverer.discover(identity, __current.ctx);
    }

    @Override
    public Driver driver() {
        return discoverer;
    }

    @Override
    public String name() {
        return name;
    }

    @Override
    public Identity identity() {
        return identity;
    }
}

<HosterAdapter.java>

/**
 * Copyright 2011 Snoopy Project

```

```

/*
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopyd.adapter;

import java.util.Map;

import Ice.Current;

import com.googlecode.snoopyd.driver.Hoster;
import com.googlecode.snoopyd.driver._IHosterDisp;

public class HosterAdapter extends _IHosterDisp {

    private Hoster hoster;

    public HosterAdapter(Hoster hoster) {
        this.hoster = hoster;
    }

    @Override
    public Map<String, String> context(Current __current) {
        return hoster.context();
    }
}

```

<ModulerAdapter.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopyd.adapter;

import java.util.Map;

import Ice.Current;

import com.googlecode.snoopyd.driver.Moduler;
import com.googlecode.snoopyd.driver._IModulerDisp;

```

```

import com.googlecode.snoopymm.ModuleNotFoundException;

public class ModulerAdapter extends _IModulerDisp {

    private Moduler moduler;

    public ModulerAdapter(Moduler moduler) {
        this.moduler = moduler;
    }

    @Override
    public Map<String, String> fetch(Current __current) {
        return moduler.fetch();
    }

    @Override
    public void deploy(String muid, String code, Current __current) {
        moduler.deploy(muid, code);
    }

    @Override
    public void undeploy(String muid, Current __current) {
        try {
            moduler.undeploy(muid);
        } catch (ModuleNotFoundException ex) {

        }
    }

    @Override
    public String[] launch(String muid, String[] params, Current __current)
        throws com.googlecode.snoopyd.driver.ModuleNotFoundException {
        return moduler.launch(muid, params);
    }

    @Override
    public void force(String muid, String[] params, Current __current) {
        moduler.force(muid, params);
    }
}

```

<SchedulerAdapter.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopyd.adapter;

import java.util.Map;

```

```

import Ice.Current;
import Ice.Identity;

import com.googlecode.snoopyd.driver.ISchedulerPrx;
import com.googlecode.snoopyd.driver.ModuleNotFoundException;
import com.googlecode.snoopyd.driver.Scheduler;
import com.googlecode.snoopyd.driver._ISchedulerDisp;

public class SchedulerAdapter extends _ISchedulerDisp {

    private Scheduler scheduler;

    public SchedulerAdapter(Scheduler scheduler) {
        this.scheduler = scheduler;
    }

    @Override
    public Map<String, String> timetable(Current __current) {
        return scheduler.timetable();
    }

    @Override
    public Map<String, String> statetable(Current __current) {
        return scheduler.statetable();
    }

    @Override
    public void schedule(String muid, long[] delays, String[] params,
                         Current __current) {
        scheduler.schedule(muid, delays, params);
    }

    @Override
    public void unschedule(String muid, Current __current)
        throws ModuleNotFoundException {
        scheduler.unschedule(muid);
    }

    @Override
    public void toogle(String muid, Current __current) {
        scheduler.toogle(muid);
    }

    @Override
    public void synchronize(Identity identity, ISchedulerPrx remoteScheduler,
                           Current __current) {
        scheduler.synchronize(identity, remoteScheduler);
    }

    @Override
    public Map<String, String> paramtable(Current __current) {
        return scheduler.paramtable();
    }

    @Override
    public void force(Identity identity, String muid, String[] params,
                     Current __current) {
        scheduler.force(identity, muid, params);
    }
}

```

<SessionierAdapter.java>

```
package com.googlecode.snoopyd.adapter;

import org.apache.log4j.Logger;

import Ice.Current;
import Ice.Identity;

import com.googlecode.snoopyd.driver.Driver;
import com.googlecode.snoopyd.driver.Sessionier;
import com.googlecode.snoopyd.driver._ISessionierDisp;
import com.googlecode.snoopyd.session.IKernelSessionPrx;
import com.googlecode.snoopyd.session.IUserSessionPrx;

public class SessionierAdapter extends _ISessionierDisp implements Adapter {

    private static Logger logger = Logger.getLogger(SessionierAdapter.class);

    private String name;
    private Ice.Identity identity;

    private Sessionier sessionier;

    public SessionierAdapter(Identity identity, Sessionier sessionier) {

        this.name = SessionierAdapter.class.getSimpleName();
        this.identity = identity;
        this.sessionier = sessionier;
    }

    @Override
    public IKernelSessionPrx createKernelSession(Identity identity,
                                                IKernelSessionPrx selfSession, Current __current) {

        return sessionier.createKernelSession(identity, selfSession);
    }

    @Override
    public IUserSessionPrx createUserSession(Identity identity,
                                             IUserSessionPrx selfSession, Current __current) {

        return sessionier.createUserSession(identity, selfSession);
    }

    @Override
    public Driver driver() {
        return sessionier;
    }

    @Override
    public String name() {
        return name;
    }

    @Override
    public Identity identity() {
        return identity;
    }
}
```

<Configuration.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.snoopyd.config;  
  
import java.io.Serializable;  
  
public class Configuration implements Serializable {  
  
    public static class ConfigurationBuilder {  
  
        private int rate = 0;  
  
        public ConfigurationBuilder rate(int rate) {  
            this.rate = rate;  
            return this;  
        }  
  
        public Configuration build() {  
            return new Configuration(rate);  
        }  
  
        }  
  
        public final int rate;  
  
        public Configuration(int rate) {  
            this.rate = rate;  
        }  
    }  

```

<ConfigurationManager.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and
```

```

 * limitations under the License.
 */
package com.googlecode.snoopyd.config;

public class ConfigurationManager {
    public ConfigurationManager() {
    }
}

```

<Identities.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopyd.util;

import Ice.Identity;

public final class Identities {

    public static Identity randomIdentity(String domain) {
        return new Identity(java.util.UUID.randomUUID().toString(), domain);
    }

    public static String toString(Identity identity) {
        return identity.category + "/" + identity.name;
    }

    public static Identity clone(Identity identity) {
        return null;
    }

    public static Identity stringToIdentity(String identity) {
        String args[] = identity.split("/");
        if (args.length > 1) {
            return new Identity(args[1], args[0]);
        } else {
            return new Identity(args[0], "");
        }
    }

    public static boolean equals(Identity id1, Identity id2) {
        return id1.name.equals(id2.name) && id1.category.equals(id2.category);
    }
}

```

```

public static Identity xor(Identity id1, Identity id2) {
    // ex: 154d2630-fafcd-4bcb-9cac-dec42ec4ba9c

    if (!id1.category.equals(id2.category)) {
        throw new IllegalArgumentException("both identities must belong one
domain");
    }

    String id1Part[] = id1.name.split("-");
    String id2Part[] = id2.name.split("-");

    String resultPart[] = new String[5];

    resultPart[0] = Long.toHexString(Long.valueOf(id1Part[0], 16) ^
Long.valueOf(id2Part[0], 16));
    while (resultPart[0].length() < 8) { resultPart[0] = "0" + resultPart[0]; }
    resultPart[1] = Long.toHexString(Long.valueOf(id1Part[1], 16) ^
Long.valueOf(id2Part[1], 16));
    while (resultPart[1].length() < 4) { resultPart[1] = "0" + resultPart[1]; }
    resultPart[2] = Long.toHexString(Long.valueOf(id1Part[2], 16) ^
Long.valueOf(id2Part[2], 16));
    while (resultPart[2].length() < 4) { resultPart[2] = "0" + resultPart[2]; }
    resultPart[3] = Long.toHexString(Long.valueOf(id1Part[3], 16) ^
Long.valueOf(id2Part[3], 16));
    while (resultPart[3].length() < 4) { resultPart[3] = "0" + resultPart[3]; }
    resultPart[4] = Long.toHexString(Long.valueOf(id1Part[4], 16) ^
Long.valueOf(id2Part[4], 16));
    while (resultPart[4].length() < 12) { resultPart[4] = "0" + resultPart[4]; }

    String result = resultPart[0] + "-" + resultPart[1] + "-" + resultPart[2] +
"-" + resultPart[3] + "-" + resultPart[4];

    return new Identity(result, id1.category);
}
}

```

<snoopyd.cmd>

```

@echo off

rem
rem Copyrighth 2011, Snoopy Project
rem

set CP=lib/snoopyd.jar;lib/log4j-
1.2.15.jar;lib/Ice.jar;config/log4j.properties;lib/sigar.jar;lib/mysql-connector-java-
5.1.16-bin.jar;
set DEFINES=-Dsnoopyd.configuration=config/snoopyd.conf -
Dlog4j.configuration=config/log4j.properties
set LIB=native/;
set ENTRY_POINT=com.googlecode.snoopyd.Launcher

java -Xms64m -Xmx128m -Xss64m -classpath %CP% -Djava.library.path=%LIB% %DEFINES% %
ENTRY_POINT%

```

<snoopyd.sh>

```
#!/bin/sh
```

```

#
# Copyrigth 2011, Snoopy Project
#
CP=lib/snoopyd.jar;lib/log4j-1.2.15.jar;lib/Ice.jar;config/log4j.properties;lib/sigar.jar;
DEFINES=-Dsnoopyd.configuration=config/snoopyd.conf
Dlog4j.configuration=config/log4j.properties
LIB=lib/;
ENTRY_POINT=com.googlecode.snoopyd.Launcher

java -Xms64m -Xmx128m -Xss64m -classpath $CP -Djava.library.path=$LIB $DEFINES
$ENTRY_POINT

<log4j.properties>

log4j.rootLogger=DEBUG, FILE, CONSOLE

log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
log4j.appender.CONSOLE.layout.ConversionPattern= %c - %m%n

log4j.appender.FILE=org.apache.log4j.RollingFileAppender
log4j.appender.FILE.File=log/snoopyd.log
log4j.appender.FILE.MaxFileSize=2MB
log4j.appender.FILE.MaxBackupIndex=2
log4j.appender.FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.FILE.layout.ConversionPattern= %c - %m%n

log4j.logger.com.googlecode.snoopyd.driver.Discoverer=DEBUG
log4j.logger.com.googlecode.snoopyd.driver.Aliver=DEBUG
log4j.logger.com.googlecode.snoopyd.driver.Networker=DEBUG
log4j.logger.com.googlecode.snoopyd.driver.Sessionier=DEBUG

```

Панель управления

<snoopycp.ice>

```

// ****
// Copyright 2011 Snoopy Project
//
// Licensed under the Apache License, Version 2.0 (the "License");
// You may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.
// ****

#ifndef SNOOPYD_ICE
#define SNOOPYD_ICE

#include <Ice/Identity.ice>

```

```

module com { module googlecode { module snoopycp {

    module core
    {

        interface IDiscoverer
        {
            void discover(Ice::Identity identity);
        };
    };

};

};

#endif

```

<Launcher.java>

```

/*
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopycp;

import com.googlecode.snoopycp.core.Snoopycp;
import javax.swing.JOptionPane;

public class Launcher {

    public static void main(String args[]) {

        int status = -1;
        Snoopycp snoopycp = new Snoopycp();
        try {
            status = snoopycp.main(Defaults.APP_NAME, args,
                System.getProperty("snoopycp.configuration",
                    Defaults.DEFAULT_CONFIGURATION));
        } catch (Ice.FileException ex) {
            System.out.println(ex.getMessage());
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, e.getMessage());
        }
        System.exit(status);
    }
}


```

<Defaults.java>

```

/*


```

```

* Copyright 2011 Snoopy Project
*
* Licensed under the Apache License, Version 2.0 (the "License");
* You may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

```

```

package com.googlecode.snoopycp;

public final class Defaults {

    public static final String APP_NAME = "snoopycp";

    public static final String APP_VER = "0.1";

    public static final String DEFAULT_CONFIGURATION = "snoopycp.conf";

    public static final String DEFAULT_ADAPTER_NAME = "Adapter";

    // path to icons, pics and other staff
    public static final String PATH_TO_SHARE = "/com/googlecode/snoopycp/share/";
}

```

<Snoopycp.java>

```

/**
 * Copyright 2011 Snoopy Project
*
* Licensed under the Apache License, Version 2.0 (the "License");
* You may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*     http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

```

```

package com.googlecode.snoopycp.core;

import com.googlecode.snoopycp.controller.Coordinator;
import com.googlecode.snoopycp.controller.DomainController;
import com.googlecode.snoopycp.ui.MainFrame;

import org.apache.log4j.Logger;

public class Snoopycp extends Ice.Application {

    public static Logger logger = Logger.getLogger(Snoopycp.class);

    public static final int EXIT_SUCCESS = 0;
}

```

```

public static final int EXIT_FAILURE = 999;

public static class ShutdownHook extends Thread {

    private Coordinator coordinator;

    public ShutdownHook(Coordinator coordinator) {
        this.coordinator = coordinator;
    }

    @Override
    public void run() {
        coordinator.terminate();
    }
}

@Override
public int run(String[] args) {

    Ice.Communicator communicator = communicator();
    Ice.Properties properties = communicator.getProperties();

    Domain domain = new Domain(communicator, properties.getProperty("Snoopy.Domain"));
    DomainController controller = new DomainController(domain);
    MainFrame view = new MainFrame(controller);
    Coordinator coordinator = new Coordinator(domain, view);

    setInterruptHook(new ShutdownHook(coordinator));

    coordinator.launch();

    return EXIT_SUCCESS;
}
}

```

<Discoverer.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopycp.core;

import com.googlecode.snoopycp.util.Identities;
import java.util.Map;
import org.apache.log4j.Logger;

```

```

public class Discoverer {

    public static Logger logger = Logger.getLogger(Discoverer.class);

    private Domain domain;

    public Discoverer(Domain domain) {
        this.domain = domain;
    }

    public void discover(Ice.Identity identity, Map<String, String> context) {
        logger.debug("discoverer received from " + Identities.toString(identity));
        domain.cacheit(identity, context);

        //System.out.println("OS type: " + context.get("os"));

    }
}

```

<DiscovererAdapter.java>

```


/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */


package com.googlecode.snoopycp.core;

import Ice.Current;
import Ice.Identity;

public class DiscovererAdapter extends _IDiscovererDisp {

    private Discoverer discoverer;

    public DiscovererAdapter(Discoverer discoverer) {
        this.discoverer = discoverer;
    }

    public void discover(Identity identity, Current __current) {
        discoverer.discover(identity, __current.ctx);
    }

}

```

<Domain.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopycp.core;

import Ice.Communicator;
import Ice.Identity;
import com.googlecode.snoopycp.Defaults;
import com.googlecode.snoopycp.util.Identities;
import com.googlecode.snoopyd.driver.IConfigurerPrx;
import com.googlecode.snoopyd.driver.IControllerPrx;
import com.googlecode.snoopyd.driver.IHosterPrx;
import com.googlecode.snoopyd.driver.IModulerPrx;
import com.googlecode.snoopyd.driver.ISchedulerPrx;
import com.googlecode.snoopyd.driver.ISessionierPrx;
import com.googlecode.snoopyd.driver.ISessionierPrxHelper;
import com.googlecode.snoopyd.session.IUserSessionPrx;
import java.util.Collections;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Observable;
import java.util.Set;
import java.util.concurrent.ConcurrentHashMap;
import org.apache.log4j.Logger;

public class Domain extends Observable implements Runnable {

    public static Logger logger = Logger.getLogger(Domain.class);
    private Ice.Identity identity;
    private Ice.Communicator communicator;
    private String name;
    private Ice.ObjectAdapter adapter;
    private Set<String> hosts;
    private Map<String, Ice.Identity> enviroment;
    private Map<Ice.Identity, Long> life;
    private Map<Ice.Identity, Map<String, String>> cache;
    private Map<Ice.Identity, IHosterPrx> hosters;
    private Map<Ice.Identity, IUserSessionPrx> sessions;
    private Map<Ice.Identity, IControllerPrx> controllers;
    private Map<Ice.Identity, Integer> hashes;
    private Map<Ice.Identity, String> osPull;
    private Map<Ice.Identity, IModulerPrx> modulers;
    private Map<Ice.Identity, ISchedulerPrx> schedulers;
    private Map<Ice.Identity, Map<String, String>> modulesStatus;
    private Map<Ice.Identity, Map<String, String>> modulesName;
    private Map<Ice.Identity, IConfigurerPrx> configurers;
}

```

```

public Domain(Communicator communicator, String name) {
    this.identity = Identities.randomIdentity(name);
    this.communicator = communicator;
    this.name = name;

    this.hosts = new HashSet<String>();
    this.environment = new HashMap<String, Ice.Identity>();

    this.life = new HashMap<Identity, Long>();

    this.cache = new ConcurrentHashMap<Ice.Identity, Map<String, String>>();

    this.sessions = new ConcurrentHashMap<Ice.Identity, IUserSessionPrx>();
    this.controllers = new ConcurrentHashMap<Ice.Identity, IControllerPrx>();
    this.hosters = new ConcurrentHashMap<Ice.Identity, IHosterPrx>();
    this.osPull = new ConcurrentHashMap<Ice.Identity, String>();
    this.modulators = new ConcurrentHashMap<Ice.Identity, IModulerPrx>();
    this.schedulers = new ConcurrentHashMap<Ice.Identity, ISchedulerPrx>();
    this.configurers = new ConcurrentHashMap<Ice.Identity, IConfigurerPrx>();
    this.modulesStatus = new ConcurrentHashMap<Ice.Identity, Map<String, String>>();
    this.modulesName = new ConcurrentHashMap<Ice.Identity, Map<String, String>>();

    this.hashes = new HashMap<Identity, Integer>();

    this.adapter = communicator.createObjectAdapter(Defaults.DEFAULT_ADAPTER_NAME);

    this.adapter.add(new DiscovererAdapter(new Discoverer(this)),
communicator.stringToIdentity(Discoverer.class.getSimpleName()));
    this.adapter.activate();

    Thread self = new Thread(this);
    // FIXME too early start
    self.start();
}

public String name() {
    return name;
}

public Ice.Identity identity() {
    return identity;
}

public Ice.Communicator communicator() {
    return communicator;
}

public void cacheit(Ice.Identity identity, Map<String, String> context) {
    synchronized (this) {

        boolean changed = false;

        int oldSize = cache.size();
        cache.put(identity, context);
        int newSize = cache.size();

        life.put(identity, System.currentTimeMillis());
    }
}

```

```

if (oldSize != newSize) {

    hosts.add(context.get("hostname"));

    enviroment.put(context.get("hostname"), identity);

    osPull.put(identity, context.get("os"));

    String proxy = Identities.toString(identity) + ":" + +
context.get("primary");
    try {
        ISessionierPrx remoteSessionier =
ISessionierPrxHelper.checkedCast(communicator.stringToProxy(proxy));
        logger.debug("Checked cast to remote session");

        remoteSessionier.ice_ping();
        IUserSessionPrx remoteSessionPrx =
remoteSessionier.createUserSession(identity(), null);
        sessions.put(identity, remoteSessionPrx);

        remoteSessionPrx.ice_ping();
        IHosterPrx remoteHoster = remoteSessionPrx.hoster();
        hosters.put(identity, remoteHoster);

        remoteSessionPrx.ice_ping();
        IControllerPrx remoteController = remoteSessionPrx.controller();
        controllers.put(identity, remoteController);

        remoteSessionPrx.ice_ping();
        IModulerPrx remoteModuler = remoteSessionPrx.moduler();
        modulers.put(identity, remoteModuler);
        remoteModuler.ice_ping();
        modulesName.put(identity, remoteModuler.fetch());

        remoteSessionPrx.ice_ping();
        ISchedulerPrx remoteScheduler = remoteSessionPrx.scheduler();
        schedulers.put(identity, remoteScheduler);

        schedulers.get(identity).ice_ping();
        modulesStatus.put(identity, schedulers.get(identity).statetable());

        remoteSessionPrx.ice_ping();
        IConfigurerPrx remoteConfigurer = remoteSessionPrx.configurer();
        configurers.put(identity, remoteConfigurer);
    } catch (Ice.ConnectionRefusedException ex) {
        logger.warn("Problem with fetch information about remote node: " +
ex.getMessage());
        // TODO do something with problem: del already added info
    }
    changed = true;
    logger.debug("New host: " + context.get("hostname") + " was added");
    logger.debug(newSize + " hosts in domain");
} else {
    if (hosts.contains(context.get("hostname")) &&
enviroment.get(context.get("hostname")) == null) {
        enviroment.put(context.get("hostname"), identity);
        changed = true;
    }
}

```

```

        if ((hashes.get(identity)) == null ? 0 : hashes.get(identity)) != context.hashCode()) {
            hashes.put(identity, context.hashCode());
            changed = true;
        }

        if (changed) {
            notifyObserver();
        }
    }
}

public Set<String> hosts() {
    return hosts;
}

public Map<Ice.Identity, String> osPull() {
    return osPull;
}

public boolean isDied(Ice.Identity identity) {
    return !enviroment.containsValue(identity);
}

public Map<String, Ice.Identity> enviroment() {
    return Collections.unmodifiableMap(enviroment);
}

public Map<String, String> cache(Ice.Identity identity) {
    return cache.get(identity);
}

public IHosterPrx hoster(Ice.Identity identity) {
    return hosters.get(identity);
}

public IModulerPrx moduler(Ice.Identity _identity) {
    return modulers.get(_identity);
}

public ISchedulerPrx scheduler(Ice.Identity _identity) {
    return schedulers.get(_identity);
}

public IControllerPrx controller(Ice.Identity identity) {
    return controllers.get(identity);
}

public IConfigurerPrx configurer(Ice.Identity identity) {
    return configurers.get(identity);
}

public Map<String, String> moduleStatus(Ice.Identity _ident) {
    return modulesStatus.get(_ident);
}

public Map<String, String> moduleName(Ice.Identity _ident) {
    return modulesName.get(_ident);
}

public IUserSessionPrx session(Ice.Identity identity) {

```

```

        return sessions.get(identity);
    }

    public void updateModules(Ice.Identity _ident) {
        try {
            this.sessions.get(_ident).ice_ping();
            HashMap<String, String> map = (HashMap<String, String>) this.sessions.get(_ident).moduler().fetch();
            this.modulesName.remove(_ident);
            this.modulesName.put(_ident, map);
            schedulers.get(_ident).ice_ping();
            modulesStatus.remove(_ident);
            modulesStatus.put(_ident, schedulers.get(_ident).statetable());
            notifyObserver();
        }

        logger.debug("Modules list updated");
    } catch (Ice.ConnectionRefusedException ex) {
        logger.warn("Updating modules list failed: " + ex.getMessage());
    }
}

@Override
public void run() {

    for (;;) {

        boolean changed = false;
        //modulesStatus.clear();
        synchronized (this) {
            for (String host : hosts) {
                Ice.Identity id = enviroment.get(host);

                if (id == null) {
                    continue;
                }

                if (System.currentTimeMillis() - life.get(id) > 10000) {
                    // FIXME Then node back to life adding new node
                    enviroment.remove(host);
                    logger.debug("Host " + host + " didn't response and was removed
from domain");
                    changed = true;
                }
            }
        }

        if (changed) {
            notifyObserver();
        }

        try {
            Thread.sleep(3000);
        } catch (InterruptedException ex) {
            logger.warn("Thread can't fall into dream: " + ex.getMessage());
        }
    }
}

```

```

    public void notifyObserver() {
        setChanged();
        notifyObservers();
        logger.debug("Domain is changed. Observers notified.");
    }

    public void removeHost(String _hostname) {
        this.hosts.remove(_hostname);
        logger.debug("Host: " + _hostname + " was removed. " + hosts.size());
        notifyObserver();
    }
}

```

<DomainController.java>

```

/*
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopycp.controller;

import com.googlecode.snoopycp.core.Domain;
import com.googlecode.snoopycp.model.GraphModel;
import com.googlecode.snoopycp.model.TableModel;
import com.googlecode.snoopycp.model.TreeModel;
import java.awt.Color;
import java.awt.Paint;
import org.apache.commons.collections15.Transformer;

public class DomainController {

    private Domain domain;

    public DomainController(Domain domain) {
        this.domain = domain;
    }

    public TableModel createTableModel() {
        return new TableModel(domain);
    }

    public TreeModel createTreeModel() {
        return new TreeModel(domain);
    }

    public GraphModel createGraphModel() {
        return new GraphModel(domain);
    }
}
```

```

public Transformer<String, String> createLabelTransformer() {
    return new Transformer<String, String>() {

        @Override
        public String transform(String vertex) {
            return vertex;
        }
    };
}

public Transformer<String, Paint> createFillTransformer() {
    return new Transformer<String, Paint>() {

        @Override
        public Paint transform(String vertex) {

            if (domain.enviroment().get(vertex) == null) {
                return Color.GRAY;
            } else {
                Ice.Identity identity = domain.enviroment().get(vertex);
                String state = domain.cache(identity).get("state");

                if (state.equals("OnlineState")) {
                    return Color.YELLOW;
                } else if (state.equals("ActiveState")) {
                    return Color.RED;
                } else {
                    return Color.BLUE;
                }
            }
        }
    };
}
}

```

<Coordinator.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopycp.controller;

import com.googlecode.listener.*;
import com.googlecode.snoopycp.Defaults;
import com.googlecode.snoopycp.ui.MainFrame;
import com.googlecode.snoopycp.core.Domain;
import java.awt.event.ActionListener;

```

```

import java.util.HashMap;
import java.util.Map;
import org.apache.log4j.Logger;

public class Coordinator {

    public static Logger logger = Logger.getLogger(Coordinator.class);
    private Domain domain;
    private MainFrame view;

    public Coordinator(Domain _domain, MainFrame view) {
        this.domain = _domain;
        this.view = view;
        this.view.setActionsOnPopup(this.packActions());
        domain.addObserver(view);
        view.setCoordinator(this);
    }

    public Domain domain() {
        return this.domain;
    }

    private Map<String, ActionListener> packActions() {
        Map<String, ActionListener> actions = new HashMap<String, ActionListener>();
        actions.put("NodeProperties", new NodePropertiesAL(view, domain));
        actions.put("ForceStart", new NodeForceStartAL(view, domain));
        actions.put("Shutdown", new NodeShutdownAL(view, domain));
        actions.put("ModuleProperties", new ModulePropertiesAL(view, domain));
        actions.put("HostResults", new HostResultsAL(view, this, logger));
        actions.put("ModuleResults", new ModuleResultsAL(view, this, logger));
        return actions;
    }

    public void launch() {
        view.setTitle("[" + domain.name() + "] " + Defaults.APP_NAME + " " +
Defaults.APP_VER);

        view.setLocationRelativeTo(null);
        view.setVisible(true);

        synchronized (this) {
            try {
                wait();
            } catch (InterruptedException ex) {
                logger.error(ex.getMessage());
            }
        }
    }

    public void terminate() {
        synchronized (this) {
            notify();
        }
    }
}

<HostResultsAL.java>

```

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.listener;

import com.googlecode.snoopycp.controller.Coordinator;
import com.googlecode.snoopycp.core.Domain;
import com.googlecode.snoopycp.model.DataBaseHostTableModel;
import com.googlecode.snoopycp.model.Node;
import com.googlecode.snoopycp.ui.MainFrame;
import com.googlecode.snoopycp.ui.Results;
import java.sql.Connection;
import java.sql.Statement;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.JOptionPane;
import javax.swing.tree.DefaultMutableTreeNode;
import org.apache.log4j.Logger;

/**
 *
 * @author Leo
 */
public class HostResultsAL implements ActionListener {

    private MainFrame view;          // View
    private Coordinator coordinator; // Controller
    private Domain domain;          // Model
    private Connection conn = null;  // connection to database
    private Statement stmt;
    private ResultSet rs;            // Results of sql script executing
    private Logger logger;          // Logger which prints to console

    public HostResultsAL(MainFrame view, Coordinator _coordinator, Logger _logger) {
        // Initialisation of data
        this.view = view;
        this.coordinator = _coordinator;
        this.domain = coordinator.domain();
        logger = _logger;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        // Get last selected node in the Tree

```

```

        DefaultMutableTreeNode      lastSelectNode      =      (DefaultMutableTreeNode)
view.getTree().getLastSelectedPathComponent();
        Node node = (Node) lastSelectNode.getUserObject();
        String[][] results = this.getHostResults(node.name); // get data from database
        if (results != null) {
            // Display results in new window
            view.addInternalFrame(new      Results(new      DataBaseHostTableModel(results),
node.name));
        } else {
            JOptionPane.showMessageDialog(view, "Cannot connect to database");
        }
    }

    public String[][] getHostResults(String hostname) {
        // SQL script
        String sql = "select Module.name, Result.`result`, Result.datestamp from `Result`,
`Host`, `Module` where Host.`name`='" + hostname + "' and Host.idHost=Result.idHost and
Module.idModule=Result.idModule";
        // Url to connect
        String                               url
domain.configurer(domain.enviroment().get(hostname)).configuration().get("connectionstring");
        logger.debug("Connect to database: " + url);
        String[][] results = null;

        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance(); // load driver
            conn = DriverManager.getConnection(url); // get connection
            logger.debug("Database connection established");
            stmt = conn.createStatement();
            if (stmt.execute(sql)) { // execute sql script
                rs = stmt.getResultSet(); // get results
                int i = 0;
                results = new String[3][];
                while (rs.next()) { // calculate row count
                    i++;
                }
                results[0] = new String[i];
                results[1] = new String[i];
                results[2] = new String[i];
                rs.beforeFirst(); // initialisation result set
                rs = stmt.getResultSet();
                i = 0;
                while (rs.next()) {
                    results[0][i] = rs.getString(1); // Fill
                    results[1][i] = rs.getString(2); // the
                    results[2][i] = rs.getString(3); // results
                    i++;
                }
            } else {
                logger.debug("SQL statement wasn't execute");
            }
        } catch (InstantiationException ex) {
            logger.error("Instantiation");
        } catch (IllegalAccessException ex) {
            logger.error("IllegalAccess");
        } catch (ClassNotFoundException ex) {
            logger.error("ClassNotFound");
        } catch (SQLException e) {
            logger.error("Cannot connect to database server");
        }
    }
}

```

```
        return results;
    }
}
```

<ModulePropertiesAL.java>

```
/***
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.listener;

import com.googlecode.snoopycp.core.Domain;
import com.googlecode.snoopycp.model.Node;
import com.googlecode.snoopycp.ui.MainFrame;
import com.googlecode.snoopycp.ui.ModulePropertyInternalFrame;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.HashMap;
import javax.swing.tree.DefaultMutableTreeNode;

public class ModulePropertiesAL implements ActionListener {

    private MainFrame view;
    private Domain domain;

    public ModulePropertiesAL(MainFrame view, Domain _domain) {
        this.view = view;
        this.domain = _domain;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        // get selected node
        DefaultMutableTreeNode lastSelectNode = (DefaultMutableTreeNode)
view.getTree().getLastSelectedPathComponent();
        // get object with information about node
        Node node = (Node) lastSelectNode.getUserObject();
        // get information about host
        HashMap<String, String> map = (HashMap<String, String>)
domain.hoster(node.identity).context();
        // put in map ip-address
        map.put("IP", domain.cache(node.identity).get("primary").split(" ")[2]);
        // display all in window
        view.addInternalFrame(new ModulePropertyInternalFrame(domain, node.identity,
node.muid));
    }
}
```

<ModuleResultsAL.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.listener;  
  
import com.googlecode.snoopycp.controller.Coordinator;  
import com.googlecode.snoopycp.core.Domain;  
import com.googlecode.snoopycp.model.DataBaseModuleTableModel;  
import com.googlecode.snoopycp.model.Node;  
import com.googlecode.snoopycp.ui.MainFrame;  
import com.googlecode.snoopycp.ui.Results;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.util.Set;  
import javax.swing.JOptionPane;  
import javax.swing.tree.DefaultMutableTreeNode;  
import org.apache.log4j.Logger;  
  
/**  
 *  
 * @author Leo  
 */  
public class ModuleResultsAL implements ActionListener {  
  
    private MainFrame view;           // View  
    private Coordinator coordinator; // Controller  
    private Domain domain;          // Model  
    private Connection conn = null;   // connection to database  
    private Statement stmt;  
    private ResultSet rs;             // Results of sql script executing  
    private Logger logger;           // Logger which prints to console  
  
    public ModuleResultsAL(MainFrame view, Coordinator _coordinator, Logger _logger) {  
        this.view = view;  
        this.coordinator = _coordinator;  
        this.logger = _logger;  
    }  
  
    @Override  
    public void actionPerformed(ActionEvent e) {
```

```

// get selected node
DefaultMutableTreeNode      lastSelectNode      =      (DefaultMutableTreeNode)
view.getTree().getLastSelectedPathComponent();
// get object with information about node
Node node = (Node) lastSelectNode.getUserObject();
logger.debug(node.identity + " " + node.muid);
// get list of hosts registered in panel
Set<String> hosts = coordinator.domain().hosts();
String hostname = null;
for (String host : hosts) { // for all hosts
    if (coordinator.domain().enviroment().get(host).equals(node.identity)) {
        hostname = host; // get host name of selected node
        break;
    }
}
// get data from database
String[][] results = this.getModuleResults(hostname, node.name);
if (results != null) {
    // Display all in window
    view.addInternalFrame(new      Results(new      DataBaseModuleTableModel(results),
node.name));
} else {
    JOptionPane.showMessageDialog(view, "Cannot connect to database");
}
}

public String[][] getModuleResults(String hostname, String moduleName) {
    // Sql script
    String sql = "select Result.`result`, Result.datestamp from `Result`, `Host`,
`Module` where Host.`name`='" + hostname + "' and Host.idHost=Result.idHost and
Module.idModule=Result.idModule and Module.`name`='" + moduleName + "'";
    // String to connect
    String                               url
domain.configurer(domain.enviroment().get(hostname)).configuration().get("connectionstring
");
    logger.debug("Connect to database: " + url);
    String[][] results = null; // results

    try {
        Class.forName("com.mysql.jdbc.Driver").newInstance(); // load driver
        conn = DriverManager.getConnection(url);                      // connect
        logger.debug("Database connection established");           // all is good
        stmt = conn.createStatement();
        if (stmt.execute(sql)) {
            rs = stmt.getResultSet(); // get results from base
            int i = 0;
            results = new String[2][];
            while (rs.next()) { // calculate row count
                i++;
            }
            results[0] = new String[i];
            results[1] = new String[i];
            rs.beforeFirst();          // reinit resultset
            rs = stmt.getResultSet();
            i = 0;
            while (rs.next()) {
                results[0][i] = rs.getString(1); // get data
                results[1][i] = rs.getString(2); // in string format
                i++;
            }
        }
    } else {
}
}

```

```

        logger.debug("SQL statement wasn't execute");
    }
} catch (InstantiationException ex) {
    logger.error("Instantiation");
} catch (IllegalAccessException ex) {
    logger.error("IllegalAccess");
} catch (ClassNotFoundException ex) {
    logger.error("ClassNotFound");
} catch (SQLException e) {
    logger.error("Cannot connect to database server");
}
return results;
}
}

```

<NodeForceStartAL.java>

```

/*
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.listener;

import com.googlecode.snoopycp.core.Domain;
import com.googlecode.snoopycp.model.Node;
import com.googlecode.snoopycp.ui.MainFrame;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.tree.DefaultMutableTreeNode;

public class NodeForceStartAL implements ActionListener {

    private MainFrame view;
    private Domain domain;

    public NodeForceStartAL(MainFrame view, Domain _domain) {
        this.view = view;
        this.domain = _domain;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        // get selected node
        DefaultMutableTreeNode lastSelectNode = (DefaultMutableTreeNode)
view.getTree().getLastSelectedPathComponent();
        // get object with information about node
        Node node = (Node) lastSelectNode.getUserObject();
        // command module to "start" with parameters
    }
}
```

```

        this.domain.moduler(node.identity).force(node.muid,
this.view.showInputDialog().split(";"));

    }
}

```

<NodePropertiesAL.java>

```

/*
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.listener;

import com.googlecode.snoopycp.core.Domain;
import com.googlecode.snoopycp.model.Node;
import com.googlecode.snoopycp.ui.MainFrame;
import com.googlecode.snoopycp.ui.NodePropertiesInternalFrame;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.HashMap;
import javax.swing.tree.DefaultMutableTreeNode;

public class NodePropertiesAL implements ActionListener {

    private MainFrame view;
    private Domain domain;

    public NodePropertiesAL(MainFrame view, Domain _domain) {
        this.view = view;
        this.domain = _domain;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        // get selected node
        DefaultMutableTreeNode lastSelectNode = (DefaultMutableTreeNode)
view.getTree().getLastSelectedPathComponent();
        // get object with information about node
        Node node = (Node) lastSelectNode.getUserObject();
        // list of node properties
        HashMap<String, String> map = (HashMap<String, String>)
domain.hoster(node.identity).context();
        // put in list ip-address
        map.put("IP", domain.cache(node.identity).get("primary").split(" ")[2]);
        // display properties in new window
        view.addInternalFrame(new NodePropertiesInternalFrame(map, node.name,
this.domain.configurer(node.identity)));
    }
}

```

```
    }
}
```

<NodeShutdownAL.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.listener;  
  
import com.googlecode.snoopycp.core.Domain;  
import com.googlecode.snoopycp.model.Node;  
import com.googlecode.snoopycp.ui.MainFrame;  
import com.googlecode.snoopycp.util.Identities;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import javax.swing.tree.DefaultMutableTreeNode;  
  
public class NodeShutdownAL implements ActionListener {  
  
    private MainFrame view;  
    private Domain domain;  
  
    public NodeShutdownAL(MainFrame view, Domain _domain) {  
        this.view = view;  
        this.domain = _domain;  
    }  
  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        // get selected node  
        DefaultMutableTreeNode lastSelectNode = (DefaultMutableTreeNode)  
view.getTree().getLastSelectedPathComponent();  
        // get object with information about node  
        Node node = (Node) lastSelectNode.getUserObject();  
        // command "shutdown" to selected node  
        domain.controller(node.identity).shutdown();  
        for (String host : domain.hosts()) { // search in all hosts  
            if (Identities.equals(domain.enviroment().get(host), node.identity)) {  
                domain.removeHost(host); // remove host from domain after shutdown  
            }  
        }  
        // Domain is changed, notify to somebody  
        domain.notifyObserver();  
    }  
}
```

<DataBaseHostTableModel.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *      http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
package com.googlecode.snoopycp.model;  
  
import javax.swing.table.AbstractTableModel;  
  
/**  
 *  
 * @author Leo  
 */  
public class DataBaseHostTableModel extends AbstractTableModel {  
  
    String[] names = {"Module name", "Result", "Date"};  
    String[][] results;  
  
    public DataBaseHostTableModel(String[][] _result) {  
        this.results = _result;  
    }  
  
    @Override  
    public String getColumnName(int column) {  
        return names[column];  
    }  
  
    @Override  
    public int getRowCount() {  
        return results[0].length;  
    }  
  
    @Override  
    public int getColumnCount() {  
        return names.length;  
    }  
  
    @Override  
    public Object getValueAt(int rowIndex, int columnIndex) {  
        return results[columnIndex][rowIndex];  
    }  
}
```

<DataBaseModuleTableModel.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");
```

```

* You may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
*      http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
package com.googlecode.snoopycp.model;

import javax.swing.table.AbstractTableModel;

public class DataBaseModuleTableModel extends AbstractTableModel {

    String[] names = {"Result", "Date"};
    String[][] results;

    public DataBaseModuleTableModel(String[][] _results) {
        this.results = _results;
    }

    @Override
    public String getColumnName(int column) {
        return names[column];
    }

    @Override
    public int getRowCount() {
        return results[0].length;
    }

    @Override
    public int getColumnCount() {
        return names.length;
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        return results[columnIndex][rowIndex];
    }
}

```

<GraphModel.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.

```

```

*/
package com.googlecode.snoopycp.model;

import com.googlecode.snoopycp.core.Domain;
import com.googlecode.snoopycp.util.Identities;
import edu.uci.ics.jung.graph.DirectedSparseGraph;
import edu.uci.ics.jung.graph.Graph;
import java.util.Map;

public class GraphModel {

    private Domain domain;
    private Graph<String, String> graph;

    public GraphModel(Domain domain) {
        this.domain = domain;

        graph = new DirectedSparseGraph<String, String>();
    }

    public void update() {

        for (String host : domain.hosts()) {

            graph.addVertex(host);

            Ice.Identity identity = domain.enviroment().get(host);

            if (identity != null) {

                Map<String, String> ctx = domain.cache(identity);

                for (String child : ctx.get("childs").split(";")) {

                    if (!child.equals("")) {

                        Ice.Identity childIdentity = Identities.stringToIdentity(child);

                        for (String childhost : domain.hosts()) {
                            if (childIdentity.equals(domain.enviroment().get(childhost)))
{
                                String edgeString = Identities.toString(身份entities.xor(identity, childIdentity));
                                graph.addEdge(edgeString, host, childhost);
                                break;
                            }
                        }
                    }
                }
            }
        }
    }

    public Graph<String, String> graph() {
        return graph;
    }
}

```

<Node.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.snoopycp.model;  
  
import Ice.Identity;  
import java.util.Map;  
  
public class Node {  
  
    public final Ice.Identity identity;  
    public final String name;  
    public String muid;  
    public Map<String, String> moduleStatuses;  
  
    public enum Type {  
        DOMEN, NODE, MODULE  
    };  
  
    public enum OsType {  
        WIN, LIN, MAC, UNIX, UNKNOWN  
    };  
    public Type nodeType;  
    public OsType os;  
  
    public Node(Identity _identity, String _name, Type _type, OsType _os) {  
        this.identity = _identity;  
        this.name = _name;  
        this.nodeType = _type;  
        this.os = _os;  
    }  
  
    public Node(Identity _identity, String _name, String _muid, Type _type, Map<String,  
String> _moduleStatuses) {  
        this(_identity, _name, _type, null);  
        muid = _muid;  
        moduleStatuses = _moduleStatuses;  
    }  
  
    @Override  
    public String toString() {  
        return name;  
    }  
}
```

```
}
```

<ParamTableModel.java>

```
/**  
 * Copyright 2011 Snoopy Project  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License");  
 * You may not use this file except in compliance with the License.  
 * You may obtain a copy of the License at  
 *  
 *     http://www.apache.org/licenses/LICENSE-2.0  
 *  
 * Unless required by applicable law or agreed to in writing, software  
 * distributed under the License is distributed on an "AS IS" BASIS,  
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 * See the License for the specific language governing permissions and  
 * limitations under the License.  
 */  
  
package com.googlecode.snoopycp.model;  
  
import com.googlecode.snoopycp.core.Domain;  
import javax.swing.table.AbstractTableModel;  
import org.apache.log4j.Logger;  
  
public class ParamTableModel extends AbstractTableModel {  
  
    Domain domain;  
    Ice.Identity ident;  
    Logger logger;  
    String muid;  
  
    public ParamTableModel(Domain _domain, Ice.Identity _ident, Logger _logger, String  
_muid) {  
        domain = _domain;  
        ident = _ident;  
        logger = _logger;  
        muid = _muid;  
    }  
  
    @Override  
    public String getColumnName(int column) {  
        return "Params";  
    }  
  
    @Override  
    public int getRowCount() {  
        try {  
            domain.scheduler(ident).ice_ping();  
            String[] strs = domain.scheduler(ident).paramtable().get(muid).split(";");  
            return strs.length;  
        } catch (Ice.ConnectionRefusedException ex) {  
            logger.warn("Can't fetch timetable from " + domain.enviroment().get(ident));  
            return 1;  
        }  
    }  
  
    @Override  
    public int getColumnCount() {
```

```

        return 1;
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        try {
            domain.scheduler(ident).ice_ping();
            String[] params = domain.scheduler(ident).paramtable().get(muid).split(";");
            return params[rowIndex];
        } catch (Ice.ConnectionRefusedException ex) {
            logger.warn("Can't fetch timetable from " + domain.enviroment().get(ident));
        }
        return "Can't fetch timetable from " + domain.enviroment().get(ident);
    }
}

```

<TableModel.java>

```

<��
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopycp.model;

import com.googlecode.snoopycp.core.Domain;
import com.googlecode.snoopycp.util.Identities;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import javax.swing.table.AbstractTableModel;

public class TableModel extends AbstractTableModel implements javax.swing.table.TableModel {

    public static final String[] COLUMNS = {"Property", "Value"};
    private Domain domain;
    private List<String> keys;
    private List<String> values;
    private int size;

    public TableModel(Domain domain) {
        super();
        this.domain = domain;
        this.keys = new ArrayList<String>();
        this.values = new ArrayList<String>();
        this.size = 0;
    }

    @Override
    public int getColumnCount() {

```

```

        return COLUMNS.length;
    }

@Override
public boolean isCellEditable(int rowIndex, int columnIndex) {
    return false;
}

@Override
public String getColumnName(int column) {
    return COLUMNS[column];
}

public int getCount() {
    return size;
}

public Object getValueAt(int rowIndex, int columnIndex) {
    if (columnIndex == 0) {
        return keys.get(rowIndex);
    } else {
        return values.get(rowIndex);
    }
}

public void update(Ice.Identity identity) {

    keys.clear();
    values.clear();

    size = 0;

    //System.out.println("asdfsdf");
    System.out.println(Identity.toString(identity));

    try {
        if (identity != null) {

            System.out.println("adsfasdf");
            domain.hoster(identity).ice_ping();

            Map<String, String> data = domain.hoster(identity).context();

            size = data.size();
            System.out.print(size + "---- ");

            for (String key : data.keySet()) {
                keys.add(key);
                System.out.print(key+": ");
                values.add(data.get(key));
                System.out.println(data.get(key));
            }
        }
    } catch (Exception ex) {
        System.out.println("TableModel.update: " + ex.getMessage());
    }
}
}

<TimeTableModel.java>
```

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopycp.model;

import com.googlecode.snoopycp.core.Domain;
import org.apache.log4j.Logger;
import javax.swing.table.AbstractTableModel;

public class TimeTableModel extends AbstractTableModel {

    Domain domain;
    Ice.Identity ident;
    Logger logger;
    String muid;

    public TimeTableModel(Domain _domain, Ice.Identity _ident, Logger _logger, String _muid) {
        domain = _domain;
        ident = _ident;
        logger = _logger;
        muid = _muid;
    }

    @Override
    public String getColumnName(int column) {
        return "Intervals";
    }

    @Override
    public int getRowCount() {
        try {
            // FIXME if dead and window of property is opened
            domain.scheduler(ident).ice_ping();
            String[] strs = domain.scheduler(ident).timetable().get(muid).split(";");
            return strs.length;
        } catch (Ice.ConnectionRefusedException ex) {
            logger.warn("Cann't fetch timetable from " + domain.enviroment().get(ident));
            return 1;
        }
    }

    @Override
    public int getColumnCount() {
        return 1;
    }

    @Override

```

```

        public Object getValueAt(int rowIndex, int columnIndex) {
            try {
                domain.scheduler(ident).ice_ping();
                String[] schedule = domain.scheduler(ident).timetable().get(muid).split(";");
                return schedule[rowIndex] + " ms";
            } catch (Ice.ConnectionRefusedException ex) {
                logger.warn("Can't fetch timetable from " + domain.enviroment().get(ident));
            }
            return "Can't fetch timetable from " + domain.enviroment().get(ident);
        }
    }
}

```

<TreeModel.java>

```

/*
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopycp.model;

import com.googlecode.snoopycp.Defaults;
import com.googlecode.snoopycp.core.Domain;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import javax.swing.ImageIcon;
import javax.swing.JMenuItem;
import javax.swing.JPopupMenu;
import javax.swing.JTree;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.DefaultTreeModel;
import javax.swing.tree.TreePath;

public class TreeModel extends DefaultTreeModel implements javax.swing.tree.TreeModel {

    private Domain domain; // container of all
    private JPopupMenu popupNode;
    private JPopupMenu popupModule;
    private JPopupMenu popupDomen;
    private JMenuItem menuItem; // template for all popup menu
    private Ice.Identity currentId; // Identity of current selected node
    private TreePath selectedPath;
```

```

private DefaultMutableTreeNode selectedNode;
private Node userObj; // User object initilation of node
private Map<String, ActionListener> actions;

public TreeModel(Domain domain) {
    super(new DefaultMutableTreeNode(domain.name())); // call constructor of
superclass
    this.domain = domain;
    //initPopup(); // build popup menu and bind actions
}

public void update(JTree _tree) {
    DefaultMutableTreeNode domainRoot = (DefaultMutableTreeNode) root;

    domainRoot.removeAllChildren(); // Clean tree

    Set<String> hosts = domain.hosts(); // get all host from cache
    String osName = null;
    Node.OsType os = Node.OsType.UNKNOWN;

    DefaultMutableTreeNode node;

    for (String host : hosts) {
        Ice.Identity identity = domain.enviroment().get(host);
        if (identity != null) {
            osName = domain.osPull().get(identity);
            if (osName.indexOf("Win") != -1) {
                os = Node.OsType.WIN;
            } else if (osName.indexOf("lin") != -1 || osName.indexOf("Lin") != -1) {
                os = Node.OsType.LIN;
            } else {
                os = Node.OsType.UNKNOWN;
            }
            node = new DefaultMutableTreeNode(new Node(identity, host, Node.Type.NODE,
os), true);
            HashMap<String, String> fullKeys = (HashMap) domain.moduleName(identity);
            Set<String> keys = fullKeys.keySet();
            for (String moduleID : keys) {
                node.add(new DefaultMutableTreeNode(
                    new Node(identity, fullKeys.get(moduleID), moduleID,
Node.Type.MODULE, domain.moduleStatus(identity)), false));
            }
        } else {
            node = new DefaultMutableTreeNode(new Node(identity, host + " [died]",
Node.Type.NODE, os), true);
        }
        domainRoot.add(node); // add node in tree
    }
    // Adding all modules of Node in the tree
}
}

public Enumeration<TreePath> getExpandedNodes(JTree _tree) {
    Enumeration<TreePath> selPaths = null;
    if (_tree.getSelectionPath() != null) {
        TreePath path = _tree.getSelectionPath().getParentPath().getParentPath();
        selPaths = _tree.getExpandedDescendants(path);
    }
    return selPaths;
}

```

```

public void setExpandedNodes (JTree _tree, Enumeration<TreePath> _paths) {
    if (_paths != null) {
        while (_paths.hasMoreElements()) {
            _tree.expandPath(_paths.nextElement());
        }
    }
}

private JPopupMenu getPopupMenu(Node.Type _type) {
    JPopupMenu popup = popupDomen;
    switch (_type) {
        case MODULE:
            popup = popupModule;
            break;
        case NODE:
            popup = popupNode;
            break;
        case DOMEN:
            popup = popupDomen;
            break;
    }

    return popup;
}

/**
 * Bind popup menu on _tree on mouse right click
 * @param _tree tree for popup menu binding
 */
public void setPopupMenu(JTree _tree, Map<String, ActionListener> _actions) {
    final JTree tree = _tree;
    actions = _actions;
    initPopup();
    tree.addMouseListener(new MouseAdapter() {

        @Override
        public void mouseReleased(MouseEvent e) {
            if (e.isPopupTrigger()) {
                int selRow = tree.getRowForLocation(e.getX(), e.getY());
                if (selRow >= 0) {
                    selectedPath = tree.getPathForLocation(e.getX(), e.getY());
                    tree.setSelectionPath(selectedPath);
                    Object obj = tree.getLastSelectedPathComponent();
                    if (obj != null) {
                        selectedNode = (DefaultMutableTreeNode) obj;
                        final Object tmp = selectedNode.getUserObject();
                        if (tmp instanceof String) {
                        } else {
                            userObj = (Node) tmp;
                            final Node.Type popupType = userObj.nodeType;
                            currentId = userObj.identity;
                            getPopupMenu(popupType).show(e.getComponent(), e.getX(),
e.getY());
                        }
                    }
                }
            }
        }
    });
    tree.addKeyListener(new KeyListener() {

```

```

@Override
public void keyTyped(KeyEvent ke) {
}

@Override
public void keyPressed(KeyEvent ke) {
    if (ke.getKeyCode() == ke.VK_M) {
        Object obj = tree.getLastSelectedPathComponent();
        if (obj != null) {
            selectedNode = (DefaultMutableTreeNode) obj;
            final Object tmp = selectedNode.getUserObject();
            Point p = tree.getLocationOnScreen();
            if (tmp instanceof String) {
            } else {
                userObj = (Node) tmp;
                final Node.Type popupType = userObj.nodeType;
                currentId = userObj.identity;
                getMenu(popupType).show(null, p.x, p.y);
            }
        }
    }
}

@Override
public void keyReleased(KeyEvent ke) {
    Object obj = tree.getLastSelectedPathComponent();
    if (obj != null) {
        selectedNode = (DefaultMutableTreeNode) obj;
        final Object tmp = selectedNode.getUserObject();
        Point p = tree.getLocationOnScreen();
        if (tmp instanceof String) {
        } else {
            userObj = (Node) tmp;
            final Node.Type popupType = userObj.nodeType;
            currentId = userObj.identity;
            getMenu(popupType).setVisible(false);
        }
    }
}
});

private void initPopup() {
    // Init popup menu for Module
    popupModule = new JPopupMenu();
    menuItem = new JMenuItem("Force start", getIcon("work.png"));
    menuItem.addActionListener(actions.get("ForceStart"));
    popupModule.add(menuItem);
    menuItem = new JMenuItem("Results", getIcon("database.png"));
    menuItem.addActionListener(actions.get("ModuleResults"));
    popupModule.add(menuItem);
    menuItem = new JMenuItem("Properties", getIcon("property.png"));
    menuItem.addActionListener(actions.get("ModuleProperties"));
    popupModule.add(menuItem);
    popupModule.setOpaque(true);
    popupModule.setLightWeightPopupEnabled(true);

    // Init popup menu for Node
    popupNode = new JPopupMenu();
    menuItem = new JMenuItem("Shutdown", getIcon("slash.png"));
    menuItem.addActionListener(actions.get("Shutdown"));
}

```

```

popupNode.add(menuItem);
menuItem = new JMenuItem("Results", get ImageIcon("database.png"));
menuItem.addActionListener(actions.get("HostResults"));
popupNode.add(menuItem);
menuItem = new JMenuItem("Properties", get ImageIcon("property.png"));
menuItem.addActionListener(actions.get("NodeProperties"));
popupNode.add(menuItem);

// Init popup menu for Domen (Not showing)
popupDomen = new JPopupMenu();
menuItem = new JMenuItem("Bla-bla");
menuItem.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent e) {
    }
});
popupDomen.add(menuItem);
menuItem = new JMenuItem("Force");
menuItem.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent e) {
    }
});
popupDomen.add(menuItem);
}

private ImageIcon get ImageIcon(String _iconName) {
    return new ImageIcon(getClass().getResource(Defaults.PATH_TO_SHARE + _iconName));
}
}

```

<AboutDialog.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

```

```

package com.googlecode.snoopycp.ui;

import com.googlecode.snoopycp.Defaults;
import javax.swing.ImageIcon;

```

```
public class AboutDialog extends javax.swing.JDialog {  
  
    /** Creates new form AboutDialog */  
    public AboutDialog(java.awt.Frame parent, boolean modal) {  
        super(parent, modal);  
        initComponents();  
        this.setTitle(Defaults.APP_NAME);  
        this.jTextArea1.setEditable(false);  
        this.setLocationRelativeTo(parent);  
    }  
  
    /** This method is called from within the constructor to  
     * initialize the form.  
     * WARNING: Do NOT modify this code. The content of this method is  
     * always regenerated by the Form Editor.  
     */  
    @SuppressWarnings("unchecked")  
    // <editor-fold defaultstate="collapsed" desc="Generated Code">  
    private void initComponents() {  
  
        jLabel1 = new javax.swing.JLabel();  
        btnCloseAbout = new javax.swing.JButton();  
        jScrollPane1 = new javax.swing.JScrollPane();  
        jTextArea1 = new javax.swing.JTextArea();  
  
        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);  
  
        jLabel1.setIcon(new  
        javax.swing.ImageIcon(getClass().getResource("/com/googlecode/snoopycp/share/about_logo_50  
0x224.jpg"))); // NOI18N  
  
        btnCloseAbout.setText("Close About");  
        btnCloseAbout.addActionListener(new java.awt.event.ActionListener() {  
            public void actionPerformed(java.awt.event.ActionEvent evt) {  
                btnCloseAboutActionPerformed(evt);  
            }  
        });  
  
        jTextArea1.setColumns(20);  
        jTextArea1.setFont(new java.awt.Font("DejaVu Sans", 0, 14)); // NOI18N  
        jTextArea1.setRows(5);  
        jTextArea1.setText("Snoopy\nDistributed monitoring  
system\nhttp://snoopy.googlecode.com");  
        jScrollPane1.setViewportView(jTextArea1);  
  
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  
        getContentPane().setLayout(layout);  
        layout.setHorizontalGroup(  
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
                .addGroup(layout.createSequentialGroup()  
                    .addGap(10, 10, 10)  
                    .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)  
                    .addGap(10, 10, 10)  
                )  
        );  
        layout.setVerticalGroup(  
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)  
                .addGap(10, 10, 10)  
        );  
    }  
}
```

```

        .addGroup(layout.createSequentialGroup()
            .addGap(212, 212, 212)
            .addComponent(btnCloseAbout)))
    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jLabel1)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 16,
Short.MAX_VALUE)
        .addComponent(btnCloseAbout)
        .addContainerGap())
);
pack();
}// </editor-fold>

private void btnCloseAboutActionPerformed(java.awt.event.ActionEvent evt) {
    this.dispose();
}

// Variables declaration - do not modify
private javax.swing.JButton btnCloseAbout;
private javax.swing.JLabel jLabel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTextArea jTextArea1;
// End of variables declaration
}

```

<ChooseModuleInternalFrame.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

```

```

package com.googlecode.snoopycp.ui;

import com.googlecode.snoopycp.core.Domain;
import javax.swing.DefaultListModel;
import javax.swing.JInternalFrame;

```

```

public class ChooseModuleInternalFrame extends javax.swing.JInternalFrame {

    private Domain domain;
    private DefaultListModel listAvailableModel = new DefaultListModel();
    private DefaultListModel listDeployedModel = new DefaultListModel();
    private NotepadInternalFrame notepad;

    /** Creates new form ChooseModuleInternalFrame */
    public ChooseModuleInternalFrame(Domain _domain, JInternalFrame _frame) {
        initComponents();
        domain = _domain;
        initList();
        this.setClosable(true);
        this.setTitle("Choose modules to deploy");
        notepad = (NotepadInternalFrame) _frame;
        this.btnOkChooseModule.setEnabled(false);
    }

    private void initList() {

        this.listAvailable.setModel(listAvailableModel);
        this.listDeployed.setModel(listDeployedModel);

        for (Object host : domain.hosts().toArray()) {
            this.listAvailableModel.addElement(host.toString());
        }

        this.listDeployedModel.clear();
        //}
    }

    /**
     * This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        listAvailable = new javax.swing.JList();
        jScrollPane2 = new javax.swing.JScrollPane();
        listDeployed = new javax.swing.JList();
        btnMoveOne = new javax.swing.JButton();
        btnMoveAll = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        btnOkChooseModule = new javax.swing.JButton();
        btnRemain = new javax.swing.JButton();
        btnRemainAll = new javax.swing.JButton();
        jCheckBox1 = new javax.swing.JCheckBox();
        jTextField1 = new javax.swing.JTextField();
        jLabel3 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jTextField2 = new javax.swing.JTextField();
        jLabel4 = new javax.swing.JLabel();

        jScrollPane1.setViewportView(listAvailable);

        
```

```

jScrollPane2.setViewportView(listDeployed);

btnMoveOne.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/com.googlecode/snoopycp/share/control.png"))
)); // NOI18N
btnMoveOne.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnMoveOneActionPerformed(evt);
    }
});

btnMoveAll.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/com.googlecode/snoopycp/share/control-
double.png"))); // NOI18N
btnMoveAll.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnMoveAllActionPerformed(evt);
    }
});

jLabel1.setText("Available");

jLabel2.setText("Deployed");

btnOkChooseModule.setText("Deploy");
btnOkChooseModule.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnOkChooseModuleActionPerformed(evt);
    }
});

btnRemain.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/com.googlecode/snoopycp/share/control-
180.png"))); // NOI18N
btnRemain.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnRemainActionPerformed(evt);
    }
});

btnRemainAll.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/com.googlecode/snoopycp/share/control-
double-180.png"))); // NOI18N
btnRemainAll.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnRemainAllActionPerformed(evt);
    }
});

jCheckBox1.setText("Activate module after deploying");

jLabel3.setText("Schedule");

jButton1.setText("Cancel");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

```



```

);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGap(61, 61, 61)
        .addComponent(btnMoveOne)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(btnMoveAll)
    .addGap(43, 43, 43)
    .addComponent(btnRemain)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(btnRemainAll))
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jLabel1)
    .addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 201, Short.MAX_VALUE)
    .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 195, javax.swing.GroupLayout.PREFERRED_SIZE)))
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addComponent(jCheckBox1)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jLabel3)
    .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
    .addComponent(jLabel4))
    .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(btnOkChooseModule)
    .addComponent(jButton1))
    .addContainerGap())
);

    pack();
}// </editor-fold>

private void btnMoveOneActionPerformed(java.awt.event.ActionEvent evt) {
    if (!this.listAvailable.isEmpty()) {
        int[] indexes = this.listAvailable.getSelectedIndices();
        for (int index : indexes) {

```

```

        this.listDeployedModel.addElement(this.listAvailableModel.getElementAt(index).toString());
            this.listAvailableModel.removeElementAt(index);
        }
        this.btnOkChooseModule.setEnabled(true);
    }
}

private void btnMoveAllActionPerformed(java.awt.event.ActionEvent evt) {
    if (!this.listAvailableModel.isEmpty()) {
        this.listDeployedModel.clear();
        for (Object host : domain.hosts().toArray()) {
            this.listDeployedModel.addElement(host.toString());
        }
        this.listAvailableModel.clear();
        this.btnOkChooseModule.setEnabled(true);
    }
}

private void btnRemainAllActionPerformed(java.awt.event.ActionEvent evt) {
    if (!this.listDeployedModel.isEmpty()) {
        this.listAvailableModel.clear();
        for (Object host : domain.hosts().toArray()) {
            this.listAvailableModel.addElement(host.toString());
        }
        this.listDeployedModel.clear();
        this.btnOkChooseModule.setEnabled(false);
    }
}

private void btnRemainActionPerformed(java.awt.event.ActionEvent evt) {
    if (!this.listDeployed.isSelectedEmpty()) {
        int[] indexes = this.listDeployed.getSelectedIndices();
        for (int index : indexes) {

this.listAvailableModel.addElement(this.listDeployedModel.getElementAt(index).toString());
            this.listDeployedModel.removeElementAt(index);
        }
        if(this.listDeployedModel.isEmpty()) {
            this.btnOkChooseModule.setEnabled(false);
        }
    }
}

private void btnOkChooseModuleActionPerformed(java.awt.event.ActionEvent evt) {
    notepad.deploy(
        this.listDeployedModel.toArray(),
        this.jCheckBox1.isSelected(),
        this.jTextField1.getText(),
        this.jTextField2.getText());
    this.dispose();
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    this.dispose();
}

private void jTextField2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

```

```

// Variables declaration - do not modify
private javax.swing.JButton btnMoveAll;
private javax.swing.JButton btnMoveOne;
private javax.swing.JButton btnOkChooseModule;
private javax.swing.JButton btnRemain;
private javax.swing.JButton btnRemainAll;
private javax.swing.JButton jButton1;
private javax.swing.JCheckBox jCheckBox1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField2;
private javax.swing.JList listAvailable;
private javax.swing.JList listDeploied;
// End of variables declaration
}

```

<IconTreeCellRenderer.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopycp.ui;

import com.googlecode.snoopycp.Defaults;
import com.googlecode.snoopycp.model.Node;
import java.awt.Component;
import java.util.HashMap;
import javax.swing.JTree;
import javax.swing.tree.DefaultTreeCellRenderer;
import javax.swing.ImageIcon;
import javax.swing.tree.DefaultMutableTreeNode;

public class IconTreeCellRenderer extends DefaultTreeCellRenderer {

    @Override
    public Component getTreeCellRendererComponent(JTree tree, Object value, boolean sel,
boolean expanded, boolean leaf, int row, boolean hasFocus) {
        String stringValue = tree.convertValueToText(value, sel, expanded, leaf, row,
hasFocus);

        this.hasFocus = hasFocus;
        setText(" " + stringValue + " ");
        if (sel) {
            setForeground(getTextSelectionColor());
        }
    }
}

```

```

    } else {
        setForeground(getTextNonSelectionColor());
    }

    DefaultMutableTreeNode node = (DefaultMutableTreeNode) value;
    if (node.getUserObject() instanceof String) {
        setIcon(getImageIcon("globe-network.png"));
    } else {
        Node userObj = (Node) node.getUserObject();
        switch (userObj.nodeType) {
            case DOMEN:
                setIcon(getImageIcon("network.png"));
                break;
            case NODE:
                if (userObj.os == Node.OsType.WIN) {
                    setIcon(getImageIcon("logo_win.jpg"));
                } else if (userObj.os == Node.OsType.LIN) {
                    setIcon(getImageIcon("logo_lin.jpg"));
                } else {
                    setIcon(getImageIcon("logo_qst.png"));
                }
                break;
            case MODULE:
                try {
                    HashMap<String, String> map = (HashMap) userObj.moduleStatuses;
                    if (map.get(userObj.muid).equalsIgnoreCase("on")) {
                        setIcon(getImageIcon("status-online.png"));
                    } else {
                        setIcon(getImageIcon("status-offline.png"));
                    }
                } catch (NullPointerException e) {
                    // JOptionPane.showMessageDialog(null, "No module status for " +
userObj.name);
                    setIcon(getImageIcon("status-offline.png"));
                }
                break;
            default:
                setIcon(getImageIcon("logo_qst.jpg"));
        }
    }

    setComponentOrientation(tree.getComponentOrientation());
    selected = sel;
    return this;
}

private ImageIcon getImageIcon(String _iconName) {
    return new ImageIcon(getClass().getResource(Defaults.PATH_TO_SHARE + _iconName));
}
}

```

<MainFrame.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0

```

```

*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/



package com.googlecode.snoopycp.ui;

import com.googlecode.snoopycp.controller.Coordinator;
import com.googlecode.snoopycp.controller.DomainController;
import com.googlecode.snoopycp.model.GraphModel;
import com.googlecode.snoopycp.model.TreeModel;
import edu.uci.ics.jung.algorithms.layout.FRLayout;
import edu.uci.ics.jung.algorithms.layout.Layout;
import edu.uci.ics.jung.visualization.VisualizationViewer;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Point;
import java.awt.event.ActionListener;
import java.util.Enumeration;
import java.util.Map;
import java.util.Observable;
import java.util.Observer;
import javax.swing.JInternalFrame;
import javax.swing.JOptionPane;
import javax.swing.tree.TreePath;
import org.apache.log4j.Logger;

public class MainFrame extends javax.swing.JFrame implements Observer {

    public static Logger logger = Logger.getLogger(MainFrame.class);
    private DomainController controller;
    private TreeModel treeModel;
    private GraphModel graphModel;
    private VisualizationViewer<String, String> visualizationViewer;
    private Layout<String, String> layout;
    private netMapIFrame nmif = new netMapIFrame();

    private Coordinator coordinator;

    /** Creates new form MainFrame */
    public MainFrame() {
        initComponents();
        this.setLocationRelativeTo(null);
    }

    public MainFrame(DomainController _controller) {
        this();

        this.controller = _controller;

        this.treeModel = controller.createTreeModel();
        this.graphModel = controller.createGraphModel();

        this.tree.setModel(treeModel);
        this.tree.setCellRenderer(new IconTreeCellRenderer());
    }
}

```

```

        this.layout = new FRLayout<String, String>(graphModel.graph(), new Dimension(200,
200));
        this.visualizationViewer = new VisualizationViewer<String, String>(layout);

this.visualizationViewer.getRenderingContext().setVertexLabelTransformer(controller.createLab
elTransformer());

this.visualizationViewer.getRenderingContext().setVertexFillPaintTransformer(controller.creat
eFillTransformer());

        visualizationViewer.setPreferredSize(new Dimension(50, 100));
        visualizationViewer.setBackground(Color.WHITE);
        nmif.add(visualizationViewer);

        update(null, null);

        pack();
    }

public void addInternalFrame(JInternalFrame _frame) {
    this.jdp.add(_frame);
    _frame.setLocation(centralPosition(_frame));
    _frame.setVisible(true);
}

public void setCoordinator(Coordinator _coordinator) {
    coordinator = _coordinator;
}

public void setActionsOnPopup(Map<String, ActionListener> _actions) {
    treeModel.setPopupMenu(this.tree, _actions);
}

public javax.swing.JTree getTree() {
    return this.tree;
}

public String showInputDialog() {
    return JOptionPane.showInputDialog(this, "Enter parameters for module:");
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jToolBar = new javax.swing.JToolBar();
    btnNotepad = new javax.swing.JButton();
    btnNetMap = new javax.swing.JButton();
    btnRefresh = new javax.swing.JButton();
    jsp = new javax.swing.JSplitPane();
    jdp = new javax.swing.JDesktopPane();
    jscp = new javax.swing.JScrollPane();
    tree = new javax.swing.JTree();
    jMenuBar = new javax.swing.JMenuBar();
    menuFile = new javax.swing.JMenu();
}

```

```

jMenuItem3 = new javax.swing.JMenuItem();
menuItemExit = new javax.swing.JMenuItem();
menuNetwork = new javax.swing.JMenu();
jMenuItem1 = new javax.swing.JMenuItem();
menuHelp = new javax.swing.JMenu();
menuItemAbout = new javax.swing.JMenuItem();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jToolBar.setRollover(true);

btnNotepad.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/com.googlecode/snoopycp/share/script--plus.png"))); // NOI18N
btnNotepad.setFocusable(false);
btnNotepad.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
btnNotepad.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
btnNotepad.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnNotepadActionPerformed(evt);
    }
});
jToolBar.add(btnNotepad);

btnNetMap.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/com.googlecode/snoopycp/share/node-select-child.png"))); // NOI18N
btnNetMap.setFocusable(false);
btnNetMap.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
btnNetMap.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
btnNetMap.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnNetMapActionPerformed(evt);
    }
});
jToolBar.add(btnNetMap);

btnRefresh.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/com.googlecode/snoopycp/share/refresh.png")));
// NOI18N
btnRefresh.setFocusable(false);
btnRefresh.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
btnRefresh.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
btnRefresh.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnRefreshActionPerformed(evt);
    }
});
jToolBar.add(btnRefresh);

jsp.setDividerLocation(180);
jsp.setRightComponent(jdp);

jscp.setViewportView(tree);

jsp.setLeftComponent(jscp);

menuFile.setText("File");

```

```

jMenuItem3.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/com.googlecode/snoopycp/share/script--plus.png"))); // NOI18N
jMenuItem3.setText("Notepad");
jMenuItem3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItem3ActionPerformed(evt);
    }
});
menuFile.add(jMenuItem3);

menuItemExit.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/com.googlecode/snoopycp/share/door-open.png"))); // NOI18N
menuItemExit.setText("Exit");
menuItemExit.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        menuItemExitActionPerformed(evt);
    }
});
menuFile.add(menuItemExit);

jMenuBar.add(menuFile);

menuNetwork.setText("Network");

jMenuItem1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/com.googlecode/snoopycp/share/node-select-child.png"))); // NOI18N
jMenuItem1.setText("view full map");
jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItem1ActionPerformed(evt);
    }
});
menuNetwork.add(jMenuItem1);

jMenuBar.add(menuNetwork);

menuHelp.setText("Help");

menuItemAbout.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/com.googlecode/snoopycp/share/home.png"))));
// NOI18N
menuItemAbout.setText("About");
menuItemAbout.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        menuItemAboutActionPerformed(evt);
    }
});
menuHelp.add(menuItemAbout);

jMenuBar.add(menuHelp);

setJMenuBar(jMenuBar);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

```

        .addComponent(jToolBar,           javax.swing.GroupLayout.DEFAULT_SIZE,      849,
Short.MAX_VALUE)
        .addComponent(jsp, javax.swing.GroupLayout.DEFAULT_SIZE, 849, Short.MAX_VALUE)
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addComponent(jToolBar, javax.swing.GroupLayout.PREFERRED_SIZE,      25,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jsp, javax.swing.GroupLayout.DEFAULT_SIZE,      545,
Short.MAX_VALUE))
    );

    pack();
}// </editor-fold>

private void menuItemAboutActionPerformed(java.awt.event.ActionEvent evt) {
    AboutDialog ad = new AboutDialog(this, true);
    ad.setVisible(true);
}

private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    // FIXME need call from Coordinator
    if (!nmif.isShowing()) {
        nmif.setClosable(true);
        nmif.setResizable(true);
        nmif.setSize(400, 400);
        nmif.setLocation(centralPosition(nmif));
        this.jdp.add(nmif);
        nmif.show();
    }
}

private void menuItemExitActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}

private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
    this.addInternalFrame(new NotepadInternalFrame(coordinator.domain()));
}

private void btnNotepadActionPerformed(java.awt.event.ActionEvent evt) {
    this.jMenuItem3ActionPerformed(evt);
}

private void btnNetMapActionPerformed(java.awt.event.ActionEvent evt) {
    this.jMenuItem1ActionPerformed(evt);
}

private void btnRefreshActionPerformed(java.awt.event.ActionEvent evt) {
    this.update(null, null);
}
/**
 * @param args the command line arguments
 */
// Variables declaration - do not modify
private javax.swing.JButton btnNetMap;
private javax.swing.JButton btnNotepad;
private javax.swing.JButton btnRefresh;
private javax.swing.JMenuBar jMenuBar;

```

```

private javax.swing.JMenuItem jMenuItem1;
private javax.swing.JMenuItem jMenuItem3;
private javax.swing.JToolBar jToolBar;
private javax.swing.JDesktopPane jdp;
private javax.swing.JScrollPane jscp;
private javax.swing.JSplitPane jsp;
private javax.swing.JMenu menuFile;
private javax.swing.JMenu menuHelp;
private javax.swing.JMenuItem menuItemAbout;
private javax.swing.JMenuItem menuItemExit;
private javax.swing.JMenu menuNetwork;
private javax.swing.JTree tree;
// End of variables declaration

@Override
public void update(Observable o, Object o1) {
    logger.debug("update ui");

    // TODO Save expand status before update and restore it after
    //Enumeration<TreePath> paths = treeModel.getExpandedNodes(tree);
    Enumeration<TreePath> selPaths = null;
    TreePath path = null;
    if (tree.getSelectionPath() != null) {
        path = tree.getSelectionPath().getParentPath();
    }
    selPaths = tree.getExpandedDescendants(path);
    treeModel.update(this.tree);
    tree.updateUI();
    //treeModel.setExpandedNodes(tree, paths);
    if (path != null) {
        tree.expandPath(path);
    }

    graphModel.update();
    visualizationViewer.updateUI();
    nmif.updatePanel();

    this.setPreferredSize(this.getSize());
    pack();
}

public Point centralPosition(JInternalFrame _frame) {
    int x = (this.jdp.getSize().width / 2) - (_frame.getSize().width / 2);
    int y = (this.jdp.getSize().height / 2) - (_frame.getSize().height / 2);
    return new Point(x, y);
}
}

```

<ModulePropertiesInternalFrame.java>

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * ModulePropertyInternalFrame.java
 *
 * Created on 18.05.2011, 17:19:45
 */
package com.googlecode.snoopycp.ui;

```

```

import com.googlecode.snoopycp.core.Domain;
import com.googlecode.snoopycp.model.ParamTableModel;
import com.googlecode.snoopycp.model.TimeTableModel;
import org.apache.log4j.Logger;

public class ModulePropertyInternalFrame extends javax.swing.JInternalFrame {

    /** Creates new form ModulePropertyInternalFrame */
    public static final Logger logger = 
Logger.getLogger(ModulePropertyInternalFrame.class);
    Domain domain;
    Ice.Identity ident;
    String muid;

    public ModulePropertyInternalFrame(Domain _domain, Ice.Identity _ident, String _muid)
{
        initComponents();
        domain = _domain;
        ident = _ident;
        muid = _muid;

        this.jTable2.setModel(new TimeTableModel(_domain, _ident, logger, _muid));
        this.jTable1.setModel(new ParamTableModel(_domain, _ident, logger, _muid));

        initStatus();
        this.setClosable(true);
        this.setTitle(domain.moduleName(_ident).get(muid) + "properties");
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        btnON = new javax.swing.JToggleButton();
        btnOFF = new javax.swing.JToggleButton();
        jLabel1 = new javax.swing.JLabel();
        jTabbedPane1 = new javax.swing.JTabbedPane();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTable2 = new javax.swing.JTable();
        jScrollPane2 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();
        jButton1 = new javax.swing.JButton();

        btnON.setText("ON");
        btnON.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnONActionPerformed(evt);
            }
        });

        btnOFF.setText("OFF");
        btnOFF.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnOFFActionPerformed(evt);
            }
        });

        
```

```

        }
    });

jLabel1.setFont(new java.awt.Font("Tahoma", 0, 14));
jLabel1.setText("Module status");

jTable2.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {null}
    },
    new String [] {
        "Title 1"
    }
));
jScrollPane1.setViewportView(jTable2);

jTabbedPane1.addTab("Schedule", jScrollPane1);

jTable1.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {null}
    },
    new String [] {
        "Title 1"
    }
));
jScrollPane2.setViewportView(jTable1);

jTabbedPane1.addTab("Parameters", jScrollPane2);

jButton1.setText("Close properties");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
    .addGap(0, 0, 0)
    .addComponent(jButton1)
    .addGap(0, 0, 0)
    .addComponent(jTabbedPane1, javax.swing.GroupLayout.Alignment.LEADING,
    javax.swing.GroupLayout.DEFAULT_SIZE, 243, Short.MAX_VALUE)
    .addGroup(layout.createSequentialGroup()
        .addGap(0, 0, 0)
        .addGroup(layout.createParallelGroup()
            .addComponent(jLabel11)
            .addGroup(layout.createSequentialGroup()
                .addGap(64, 64, 64)
                .addComponent(btnON, javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            )
            .addGroup(layout.createSequentialGroup()
                .addGap(18, 18, 18)
                .addComponent(btnOFF)))
        )
        .addGap(0, 0, 0)
        .addComponent(jLabel1)
        .addGap(64, 64, 64)
        .addComponent(btnOFF, javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE)
    )
    .addGap(0, 0, 0)
    .addComponent(btnON, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE)
    .addGap(18, 18, 18)
    .addComponent(btnOFF))
)
    .addGap(0, 0, 0)
    .addComponent(jLabel1)
    .addGap(64, 64, 64)
    .addComponent(btnOFF, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE)
)
);
layout.setVerticalGroup(
    layout.createParallelGroup()
        .addComponent(jButton1)
        .addComponent(jTabbedPane1)
        .addComponent(jLabel11)
        .addGroup(layout.createSequentialGroup()
            .addGap(64, 64, 64)
            .addComponent(btnON, javax.swing.GroupLayout.PREFERRED_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGroup(layout.createSequentialGroup()
                .addGap(18, 18, 18)
                .addComponent(btnOFF)))
        )
        .addComponent(jLabel1)
        .addGap(64, 64, 64)
        .addComponent(btnOFF, javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE)
)
);
layout.setHorizontalGroup(
    layout.createParallelGroup()
        .addComponent(jButton1)
        .addComponent(jTabbedPane1)
        .addComponent(jLabel11)
        .addGroup(layout.createSequentialGroup()
            .addGap(64, 64, 64)
            .addComponent(btnON, javax.swing.GroupLayout.PREFERRED_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGroup(layout.createSequentialGroup()
                .addGap(18, 18, 18)
                .addComponent(btnOFF)))
        )
        .addComponent(jLabel1)
        .addGap(64, 64, 64)
        .addComponent(btnOFF, javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE)
)
);

```

```

    .addGap(19, 19, 19)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jLabel1)
    .addComponent(btnON)
    .addComponent(btnOFF))
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addComponent(jTabbedPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 190,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(jButton1)
    .addContainerGap(26, Short.MAX_VALUE))
);

pack();
}// </editor-fold>

private void btnOFFActionPerformed(java.awt.event.ActionEvent evt) {
    if (this.btnON.isSelected()) {
        this.btnOFF.setSelected(true);
        this.btnON.setSelected(false);
    }
    // TODO if node is dead
    this.domain.scheduler(ident).toogle(muid);
    domain.updateModules(ident);
}

private void btnONActionPerformed(java.awt.event.ActionEvent evt) {
    if (this.btnOFF.isSelected()) {
        this.btnON.setSelected(true);
        this.btnOFF.setSelected(false);
    }
    this.domain.scheduler(ident).toogle(muid);
    domain.updateModules(ident);
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    this.dispose();
}
// Variables declaration - do not modify
private javax.swing.JToggleButton btnOFF;
private javax.swing.JToggleButton btnON;
private javax.swing.JButton jButton1;
private javax.swing.JLabel jLabel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JTabbedPane jTabbedPane1;
private javax.swing.JTable jTable1;
private javax.swing.JTable jTable2;
// End of variables declaration

private void initStatus() {
    String status = this.domain.scheduler(ident).statetable().get(muid);
    if (status.equalsIgnoreCase("ON")) {
        this.btnON.setSelected(true);
        this.btnOFF.setSelected(false);
    } else {
        this.btnON.setSelected(false);
        this.btnOFF.setSelected(true);
    }
}

```

```

    }

<NodePropertiesInternalFrame.java>

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopycp.ui;

import com.googlecode.snoopyd.driver.IConfigurerPrx;
import java.util.HashMap;
import java.util.Map;
import javax.swing.JOptionPane;

public class NodePropertiesInternalFrame extends javax.swing.JInternalFrame {

    IConfigurerPrx config;

    /** Creates new form NodeInfoInternalFrame */
    // FIXME change way of transfer IConfigurerPrx
    public NodePropertiesInternalFrame(Map<String, String> _info, String _nodeName, IConfigurerPrx _config) {
        initComponents();
        this.setClosable(true);
        this.setTitle(_nodeName + " - properties");
        initInfo(_info);
        config = _config;
    }

    public NodePropertiesInternalFrame(Map<String, String> _info) {
        initComponents();
        this.setClosable(true);
        this.setTitle("Properties");
        initInfo(_info);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        btnCloseProperties = new javax.swing.JButton();

```

```

jTabbedPane1 = new javax.swing.JTabbedPane();
jPanel1 = new javax.swing.JPanel();
jLabel7 = new javax.swing.JLabel();
textAddressString = new javax.swing.JTextField();
btnDataSourceConfirm = new javax.swing.JButton();
jlExamplDataSourceStrting = new javax.swing.JLabel();
btnViewCurrent = new javax.swing.JButton();
jLabel8 = new javax.swing.JLabel();
jLabel9 = new javax.swing.JLabel();
textPort = new javax.swing.JTextField();
textBasename = new javax.swing.JTextField();
jLabel10 = new javax.swing.JLabel();
textUsername = new javax.swing.JTextField();
jLabel11 = new javax.swing.JLabel();
textPassword = new javax.swing.JPasswordField();
jPanel2 = new javax.swing.JPanel();
jLabel1 = new javax.swing.JLabel();
jLabel12 = new javax.swing.JLabel();
jLabel13 = new javax.swing.JLabel();
jLabel14 = new javax.swing.JLabel();
jLabel15 = new javax.swing.JLabel();
jLabel16 = new javax.swing.JLabel();
jlCPU = new javax.swing.JLabel();
jlVendor = new javax.swing.JLabel();
jlCores = new javax.swing.JLabel();
jlFrequency = new javax.swing.JLabel();
jlRAM = new javax.swing.JLabel();
jSeparator1 = new javax.swing.JSeparator();
jLabel12 = new javax.swing.JLabel();
jLabel13 = new javax.swing.JLabel();
jLabel14 = new javax.swing.JLabel();
jLabel15 = new javax.swing.JLabel();
jLabel16 = new javax.swing.JLabel();
jlHostname = new javax.swing.JLabel();
jlIP = new javax.swing.JLabel();
jlGateway = new javax.swing.JLabel();
jlPrimDNS = new javax.swing.JLabel();

setTitle("Properties");

btnCloseProperties.setText("Close properties");
btnCloseProperties.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnClosePropertiesActionPerformed(evt);
    }
});

jLabel7.setText("Data source:");

btnDataSourceConfirm.setText("Confirm");
btnDataSourceConfirm.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnDataSourceConfirmActionPerformed(evt);
    }
});

jlExamplDataSourceStrting.setText("Basename");

btnViewCurrent.setText("View current");
btnViewCurrent.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {

```



```

        .addGroup(jPanel1Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jLabel17)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(textPort, javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(jLabel18)
            .addComponent(textAddressString,
                javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(jLabel19))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jlExamplDataSourceStrting)
            .addComponent(textBasename, javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)

        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel10)
            .addComponent(textUsername, javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)

        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel11)
            .addComponent(textPassword, javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 90,
                Short.MAX_VALUE)

        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(btnDataSourceConfirm)
            .addComponent(btnViewCurrent))
            .addContainerGap())
    );

    jTabbedPane1.addTab("Configure", jPanel1);

    jLabel1.setFont(new java.awt.Font("Tahoma", 1, 11));
    jLabel1.setText("Hardware info");

    jLabel2.setText("CPU");

    jLabel3.setText("Core");

    jLabel4.setText("Frequency");

    jLabel5.setText("RAM");

    jLabel6.setText("Vendor");

    jlCPU.setText("jLabel7");

    jlVendor.setText("jLabel8");

    jlCores.setText("jLabel9");

```



```

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jLabel16)
        .addComponent(jLabel13)
        .addComponent(jLabel14)
        .addComponent(jLabel15))
    .addGap(28, 28, 28)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jlPrimDNS)
        .addComponent(jlGateway)
        .addComponent(jlIP)
        .addComponent(jlHostname)))
    .addContainerGap(10, Short.MAX_VALUE))
);

jPanel2Layout.setVerticalGroup(
    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel2Layout.createSequentialGroup()
        .addContainerGap()
    )
    .addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addComponent(jLabel11)
        )
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addComponent(jLabel12)
        )
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addComponent(jLabel16)
        )
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addComponent(jLabel13)
        )
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addComponent(jLabel14)
        )
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addComponent(jLabel15))
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addComponent(jlCPU)
        )
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addComponent(jlVendor)
        )
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addComponent(jlCores)
            .addGap(20, 20, 20)
            .addComponent(jlFrequency))
        )
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addComponent(jlRAM))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jLabel12)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

```

```

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addComponent(jLabel16)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel13)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel14)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel15)
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addComponent(jlHostname)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jlIP)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jlGateway)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jlPrimDNS))
        .addContainerGap(41, Short.MAX_VALUE))
);

jTabbedPane1.addTab("NodeInfo", jPanel2);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(199, Short.MAX_VALUE)
            .addComponent(btnCloseProperties)
            .addGap(0, 0, 0)
            .addGroup(layout.createSequentialGroup()
                .addGap(0, 0, 0)
                .addComponent(jTabbedPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 299,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(11, Short.MAX_VALUE)))
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
            layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jTabbedPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 310,
                    Short.MAX_VALUE)
                .addGap(0, 0, 0)
                .addGroup(layout.createSequentialGroup()
                    .addGap(0, 0, 0)
                    .addComponent(btnCloseProperties)
                    .addGap(0, 0, 0)))
);
);

jTabbedPane1.getAccessibleContext().setAccessibleName("Configure");
jTabbedPane1.getAccessibleContext().setAccessibleDescription("Configure");

pack();
}// </editor-fold>

```

```

private void btnClosePropertiesActionPerformed(java.awt.event.ActionEvent evt) {
    this.dispose();
}

private void btnDataSourceConfirmActionPerformed(java.awt.event.ActionEvent evt) {
    if (!this.textAddressString.getText().equalsIgnoreCase("") &&
!this.textPort.getText().equalsIgnoreCase("") &&
!this.textUsername.getText().equalsIgnoreCase("")) {
        Map<String, String> map = new HashMap<String, String>();
        //String sourceString = this.textDataSourceString.getText();
        String address = this.textAddressString.getText();
        String port = this.textPort.getText();
        String basename = this.textBasename.getText();
        String username = this.textUsername.getText();
        String password = new String(this.textPassword.getPassword());
        String url = "jdbc:mysql://" + address + ":" + port + "/" +
                     basename + "?user=" + username + "&password=" + password;
        map.put("connectionstring", url);
        try {
            config.ice_ping();
            config.reconfigure(map);
        } catch (Ice.ConnectionRefusedException e) {
            JOptionPane.showInternalMessageDialog(this, "Node not available more " +
+ e.getMessage());
        }
    } else {
        JOptionPane.showInternalMessageDialog(this, "Some field is empty");
    }
}

private void btnViewCurrentActionPerformed(java.awt.event.ActionEvent evt) {
    String string = null;
    try {
        string = config.configuration().get("connectionstring");
        String[] strings = string.split("/");
        String address = strings[2];
        String[] tmp = strings[3].split "=";
        String basename = tmp[0].substring(0, tmp[0].length() - 5);
        String user = tmp[1].split "& "[0];
        String pass = tmp[2];
        String str = "\naddress: " + address + "\nbasename: " + basename
                    + "\nusername: " + user + "\npassword: " + pass;
        config.ice_ping();
        JOptionPane.showInternalMessageDialog(this, str, "Current data source string",
JOptionPane.INFORMATION_MESSAGE);
    } catch (Ice.ConnectionRefusedException e) {
        JOptionPane.showInternalMessageDialog(this, "Node not available more " +
e.getMessage());
    } catch (ArrayIndexOutOfBoundsException e) {
        JOptionPane.showInternalMessageDialog(this, string, "Current data source
string", JOptionPane.INFORMATION_MESSAGE);
    }
}
// Variables declaration - do not modify
private javax.swing.JButton btnCloseProperties;
private javax.swing.JButton btnDataSourceConfirm;
private javax.swing.JButton btnViewCurrent;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;

```

```

private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel13;
private javax.swing.JLabel jLabel14;
private javax.swing.JLabel jLabel15;
private javax.swing.JLabel jLabel16;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JTabbedPane jTabbedPane1;
private javax.swing.JLabel jlCPU;
private javax.swing.JLabel jlCores;
private javax.swing.JLabel jlExampleDataSourceString;
private javax.swing.JLabel jlFrequency;
private javax.swing.JLabel jlGateway;
private javax.swing.JLabel jlHostname;
private javax.swing.JLabel jlIP;
private javax.swing.JLabel jlPrimDNS;
private javax.swing.JLabel jlRAM;
private javax.swing.JLabel jlVendor;
private javax.swing.JTextField jTextFieldAddressString;
private javax.swing.JTextField jTextFieldBasename;
private javax.swing.JPasswordField jTextFieldPassword;
private javax.swing.JTextField jTextFieldPort;
private javax.swing.JTextField jTextFieldUsername;
// End of variables declaration

private void initInfo(Map<String, String> _info) {
    this.jlCPU.setText(_info.get("Model"));// FIXME may be very long name
    this.jlCores.setText(_info.get("TotalCores"));
    this.jlVendor.setText(_info.get("Vendor"));
    this.jlFrequency.setText(_info.get("Mhz") + " MHz");
    this.jlRAM.setText(_info.get("Free").substring(0, 3) + " / " + _info.get("Ram") +
" MB");
    this.jlHostname.setText(_info.get("HostName"));
    this.jlIP.setText(_info.get("IP"));
    this.jlGateway.setText(_info.get("DefaultGateway"));
    this.jlPrimDNS.setText(_info.get("PrimaryDns"));
}
}

```

<NotepadInternalFrame.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software

```

```

* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
package com.googlecode.snoopycp.ui;

import com.googlecode.snoopycp.core.Domain;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import org.apache.log4j.Logger;

public class NotepadInternalFrame extends javax.swing.JInternalFrame {

    private Logger logger = Logger.getLogger(NotepadInternalFrame.class);
    private JTextArea editArea;
    private JFileChooser fileChooser = new JFileChooser();
    //... Create actions for menu items, buttons, ...
    private Action openAction = new OpenAction();
    private Action saveAction = new SaveAction();
    private Action exitAction = new ExitAction(this);
    private Domain domain;

    //===== constructor
    public NotepadInternalFrame(Domain _domain) {
        initComponents();
        domain = _domain;
        //... Create scrollable text area.
        editArea = new JTextArea(15, 80);
        editArea.setBorder(BorderFactory.createEmptyBorder(2, 2, 2, 2));
        editArea.setFont(new Font("monospaced", Font.PLAIN, 14));
        //System.out.println("editArea are configured");
        JScrollPane scrollingText = new JScrollPane(editArea);
        //System.out.println("editArea added to scrollpane");
        //this.jScrollPane1.setViewportView(editArea);

        //-- Create a content pane, set layout, add component.
        JPanel content = new JPanel();
        content.setLayout(new BorderLayout());
        content.add(scrollingText, BorderLayout.CENTER);

        //... Create menubar
        this.menuItemOpen.setAction(openAction);
        this.menuItemSave.setAction(saveAction);
        this.menuItemClose.setAction(exitAction);

        //... Set window content and menu.
        setContentPane(content);

        //... Set other window characteristics.
        //setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle("Notepad");
        pack();
        // setLocationRelativeTo(null);
        setClosable(true);
        setResizable(true);
        setMaximizable(true);
    }
}

```

```

    public void deploy(Object[] hosts, boolean _activate, String _schedule, String
_params) {
    String hos = null;
    try {
        for (Object host : hosts) {
            hos = (String) host;
            Ice.Identity ident = domain.enviroment().get(hos);
            String muid = java.util.UUID.randomUUID().toString();
            domain.moduler(ident).ice_ping();
            domain.moduler(ident).deploy(muid, editArea.getText());
            String[] scheduleArray = _schedule.split(";");
            int[] delay = new int[scheduleArray.length];
            long[] delays = new long[scheduleArray.length];
            for (int i = 0; i < scheduleArray.length; i++) {
                delay[i] = Integer.parseInt(scheduleArray[i]);
                delays[i] = (long) delay[i] * 1000;
            }
            domain.scheduler(ident).ice_ping();
            domain.scheduler(ident).schedule(muid, delays, _params.split(";"));
            if (!_activate) {
                domain.scheduler(ident).toogle(muid);
            }
            domain.updateModules(ident);
        }
    } catch (Ice.ConnectionRefusedException e) {
        logger.warn("Problem with deploy on remote node " + hos + e.getMessage());
    }
}

/////////////////////////////// inner class OpenAction
class OpenAction extends AbstractAction {
    //===== constructor

    public OpenAction() {
        super("Open...");
        putValue(MNEMONIC_KEY, new Integer('O'));
    }

    //===== actionPerformed
    @Override
    public void actionPerformed(ActionEvent e) {
        int retval = fileChooser.showOpenDialog(NotePadInternalFrame.this);
        if (retval == JFileChooser.APPROVE_OPTION) {
            File f = fileChooser.getSelectedFile();
            try {
                FileReader reader = new FileReader(f);
                editArea.read(reader, ""); // Use TextComponent read
            } catch (IOException ioex) {
                System.out.println(e);
                System.exit(1);
            }
        }
    }
}

/////////////////////////////// inner class SaveAction
class SaveAction extends AbstractAction {
    //===== constructor

    SaveAction() {

```

```

        super("Save...");
        putValue(MNEMONIC_KEY, new Integer('S'));
    }

//===== actionPerformed
@Override
public void actionPerformed(ActionEvent e) {
    int retval = fileChooser.showSaveDialog(NotepadInternalFrame.this);
    if (retval == JFileChooser.APPROVE_OPTION) {
        File f = fileChooser.getSelectedFile();
        try {
            FileWriter writer = new FileWriter(f);
            editArea.write(writer); // Use TextComponent write
        } catch (IOException ioex) {
            JOptionPane.showMessageDialog(NotepadInternalFrame.this, ioex);
            //System.exit(1);
        }
    }
}

//////////////////////////////////////////////////////////////// inner class ExitAction
class ExitAction extends AbstractAction {

    JInternalFrame frame;
//===== constructor

    private ExitAction(JInternalFrame aThis) {
        super("Exit");
        frame = aThis;
        putValue(MNEMONIC_KEY, new Integer('X'));
    }

//===== actionPerformed
@Override
public void actionPerformed(ActionEvent e) {
    frame.dispose();
}
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jScrollPane1 = new javax.swing.JScrollPane();
    menuBar = new javax.swing.JMenuBar();
    menuFile = new javax.swing.JMenu();
    menuItemOpen = new javax.swing.JMenuItem();
    menuItemSave = new javax.swing.JMenuItem();
    jSeparator2 = new javax.swing.JPopupMenu.Separator();
    menuItemDeploy = new javax.swing.JMenuItem();
    jSeparator1 = new javax.swing.JPopupMenu.Separator();
    menuItemClose = new javax.swing.JMenuItem();

    menuFile.setText("File");
}

```

```

menuItemOpen.setText("Open");
menuFile.add(menuItemOpen);

menuItemSave.setText("Save");
menuFile.add(menuItemSave);
menuFile.add(jSeparator2);

menuItemDeploy.setText("Deploy");
menuItemDeploy.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        menuItemDeployActionPerformed(evt);
    }
});
menuFile.add(menuItemDeploy);
menuFile.add(jSeparator1);

menuItemClose.setText("Exit");
menuFile.add(menuItemClose);

menuBar.add(menuFile);

setJMenuBar(menuBar);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
    .addGap(0, 0, Short.MAX_VALUE)
    .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 374,
Short.MAX_VALUE)
    .addGap(0, 0, Short.MAX_VALUE)
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
    .addGap(0, 0, Short.MAX_VALUE)
    .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 231,
Short.MAX_VALUE)
    .addGap(0, 0, Short.MAX_VALUE)
);
}

pack();
}// </editor-fold>

private void menuItemDeployActionPerformed(java.awt.event.ActionEvent evt) {
    ChooseModuleInternalFrame cmif = new ChooseModuleInternalFrame(domain, this);
    this.getParent().add(cmif);
    cmif.show();
}
// Variables declaration - do not modify
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JPopupMenu.Separator jSeparator1;
private javax.swing.JPopupMenu.Separator jSeparator2;
private javax.swing.JMenuBar menuBar;
private javax.swing.JMenu menuFile;
private javax.swing.JMenuItem menuItemClose;
private javax.swing.JMenuItem menuItemDeploy;
private javax.swing.JMenuItem menuItemOpen;

```

```

    private javax.swing.JMenuItem menuItemSave;
    // End of variables declaration
}

<Results.java>

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopycp.ui;

import javax.swing.table.AbstractTableModel;

public class Results extends javax.swing.JInternalFrame {

    /** Creates new form Results */
    public Results(AbstractTableModel _tm, String _name) {
        initComponents();
        this.jTable1.setModel(_tm);
        this.setTitle(_name + " results");
        this.setClosable(true);
        this.setResizable(true);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();
        btnClose = new javax.swing.JButton();

        jTable1.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null}
            },
            new String [] {
                "Title 1", "Title 2", "Title 3", "Title 4"
            }
        ));
    }
}

```

```

        }
    );
    jScrollPane1.setViewportView(jTable1);

    btnClose.setText("Close results");
    btnClose.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            btnCloseActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(0, 0, Short.MAX_VALUE)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 385,
Short.MAX_VALUE)
            .addGap(0, 0, Short.MAX_VALUE)
            .addGroup(layout.createSequentialGroup()
                .addGap(0, 0, Short.MAX_VALUE)
                .addComponent(btnClose)
                .addGap(0, 0, Short.MAX_VALUE)
            )
        );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(0, 0, Short.MAX_VALUE)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 241,
Short.MAX_VALUE)
            .addGap(0, 0, Short.MAX_VALUE)
            .addComponent(btnClose)
            .addGap(0, 0, Short.MAX_VALUE)
        )
    );
    pack();
} // </editor-fold>

private void btnCloseActionPerformed(java.awt.event.ActionEvent evt) {
    this.dispose();
}

// Variables declaration - do not modify
private javax.swing.JButton btnClose;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTable1;
// End of variables declaration
}

```

<NetMapIFrame.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at

```

```

/*
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.googlecode.snoopycp.ui;

import edu.uci.ics.jung.visualization.VisualizationViewer;
import java.awt.GridLayout;

public class netMapIFrame extends javax.swing.JInternalFrame {

    /** Creates new form netMapIFrame */
    public netMapIFrame() {
        initComponents();
        this.setClosable(true);
        this.setResizable(true);
        this.setMaximizable(true);
        this.setTitle("Network Map");
        this.graphPanel.setLayout(new GridLayout(1, 1));
    }

    public void add(VisualizationViewer<String, String> _add){
        this.graphPanel.add(_add);
    }

    public void updatePanel() {
        this.graphPanel.updateUI();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jscp = new javax.swing.JScrollPane();
        graphPanel = new javax.swing.JPanel();

        javax.swing.GroupLayout graphPanelLayout = new javax.swing.GroupLayout(graphPanel);
        graphPanel.setLayout(graphPanelLayout);
        graphPanelLayout.setHorizontalGroup(
            graphPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(graphPanelLayout.createSequentialGroup()
                    .addGap(0, 390, Short.MAX_VALUE)
                    .addComponent(jscp, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        );
        graphPanelLayout.setVerticalGroup(
            graphPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jscp, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        );

        graphPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(graphPanelLayout.createSequentialGroup()
                .addGap(0, 270, Short.MAX_VALUE)
                .addComponent(jscp, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        );
        graphPanelLayout.setVerticalGroup(
            graphPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jscp, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        );
    }
}

```

```

        jscp.setViewportView(graphPanel);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jscp, javax.swing.GroupLayout.DEFAULT_SIZE,
394,
Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jscp, javax.swing.GroupLayout.DEFAULT_SIZE,
274,
Short.MAX_VALUE)
        );

        pack();
} // </editor-fold>
// Variables declaration - do not modify
private javax.swing.JPanel graphPanel;
private javax.swing.JScrollPane jscp;
// End of variables declaration
}

```

<Identities.java>

```

/**
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.googlecode.snoopycp.util;

import Ice.Identity;

public final class Identities {

    /**
     * Give you identity with random UID like 154d2630-fafdf-4bcb-9cac-dec42ec4ba9c
     * and prefix <i>domain</i>
     * @param domain category of identity
     * @return identity object
     */
    public static Identity randomIdentity(String domain) {
        return new Identity(java.util.UUID.randomUUID().toString(), domain);
    }

    /**
     * Give you full string view of identity like
     * dev/154d2630-fafdf-4bcb-9cac-dec42ec4ba9c
     * @param identity object to transfer to string
     */
}

```

```

* @return string view of <i>identity</i>
*/
public static String toString(Identity identity) {
    return identity.category + "/" + identity.name;
}

public static Identity stringToIdentity(String identity) {
    String args[] = identity.split("/");
    if (args.length > 1) {
        return new Identity(args[1], args[0]);
    } else {
        return new Identity(args[0], "");
    }
}

/**
 * Compare two identity
 * @param id1 first object to compare
 * @param id2 second object to compare
 * @return true if objects are equal or false
 */
public static boolean equals(Identity id1, Identity id2) {
    return id1.name.equals(id2.name) && id1.category.equals(id2.category);
}

public static Identity xor(Identity id1, Identity id2) {
    // ex: 154d2630-faf0-4bcb-9cac-dec42ec4ba9c

    String id1Part[] = id1.name.split("-");
    String id2Part[] = id2.name.split("-");

    String resultPart[] = new String[5];

    resultPart[0] = Long.toHexString(Long.valueOf(id1Part[0], 16) ^ Long.valueOf(id2Part[0], 16));
    while (resultPart[0].length() < 8) {
        resultPart[0] = "0" + resultPart[0];
    }
    resultPart[1] = Long.toHexString(Long.valueOf(id1Part[1], 16) ^ Long.valueOf(id2Part[1], 16));
    while (resultPart[1].length() < 4) {
        resultPart[1] = "0" + resultPart[1];
    }
    resultPart[2] = Long.toHexString(Long.valueOf(id1Part[2], 16) ^ Long.valueOf(id2Part[2], 16));
    while (resultPart[2].length() < 4) {
        resultPart[2] = "0" + resultPart[2];
    }
    resultPart[3] = Long.toHexString(Long.valueOf(id1Part[3], 16) ^ Long.valueOf(id2Part[3], 16));
    while (resultPart[3].length() < 4) {
        resultPart[3] = "0" + resultPart[3];
    }
    resultPart[4] = Long.toHexString(Long.valueOf(id1Part[4], 16) ^ Long.valueOf(id2Part[4], 16));
    while (resultPart[4].length() < 12) {
        resultPart[4] = "0" + resultPart[4];
    }
}

```

```

        String result = resultPart[0] + "-" + resultPart[1] + "-" + resultPart[2] + "-" +
resultPart[3] + "-" + resultPart[4];

        return new Identity(result, id1.category);
    }
}

<log4j.properties>

log4j.rootLogger=DEBUG, CONSOLE

log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
log4j.appender.CONSOLE.layout.ConversionPattern= %c - %m%n

```

```

log4j.logger.com.googlecode.snoopyd.driver.Discoverer=DEBUG
log4j.logger.com.googlecode.snoopyd.driver.Aliver=DEBUG
log4j.logger.com.googlecode.snoopyd.driver.Networker=WARN
log4j.logger.com.googlecode.snoopyd.driver.Sessionier=DEBUG

```

<gendb-mysql.sql>

```

/*
 * Copyright 2011 Snoopy Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * You may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
*/

```

```
Drop database If Exists snoopydb;
```

```
Create Database snoopydb Character Set utf8 Collate utf8_general_ci;
```

```
Use snoopydb;
```

```
Create table Module (
    idModule Int NOT NULL AUTO_INCREMENT,
    name Varchar(128),
    Primary Key (idModule)) ENGINE = InnoDB;
```

```
Create table Result (
    idResult Int NOT NULL AUTO_INCREMENT,
    idHost Int NOT NULL,
    idModule Int NOT NULL,
    datestamp Date,
    result Varchar(512),
    Primary Key (idResult)) ENGINE = InnoDB;
```

```
Create table Host (
    idHost Int NOT NULL AUTO_INCREMENT,
```

```

    idOs Int NOT NULL,
    name Varchar(128),
Primary Key (idHost) ENGINE = InnoDB;

Create table Os (
    idOs Int NOT NULL AUTO_INCREMENT,
    name Varchar(128),
Primary Key (idOs) ENGINE = InnoDB;

Alter table Result add Foreign Key (idModule) references Module (idModule) on delete cascade on update restrict;
Alter table Result add Foreign Key (idHost) references Host (idHost) on delete cascade on update restrict;
Alter table Host add Foreign Key (idOs) references Os (idOs) on delete restrict on update restrict;

Delimiter $$

Create Procedure storeResult(In osname Varchar(128), In hostname Varchar(128), In modulename Varchar(128), In result Varchar(512))
Begin
    Declare osIdCount Int;
    Declare hostIdCount Int;
    Declare moduleIdCount Int;

    Declare osId Int;
    Declare hostId Int;
    Declare moduleId Int;

    Select Count(*) Into osIdCount From Os Where name=osname;
    If osIdCount = 0 Then
        Insert Into Os(name) Values(osname);
        Set osId = LAST_INSERT_ID();
    Else
        Select idOs Into osId From Os Where name=osname;
    End If;

    Select Count(*) Into hostIdCount From Host Where name=hostname;
    If hostIdCount = 0 Then
        Insert Into Host(idOs, name) Values(osId, hostname);
        Set hostId = LAST_INSERT_ID();
    Else
        Select idHost Into hostId From Host Where name=hostname;
    End If;

    Select Count(*) Into moduleIdCount From Module Where name=modulename;
    If moduleIdCount = 0 Then
        Insert Into Module(name) Values(modulename);
        Set moduleId = LAST_INSERT_ID();
    Else
        Select idModule Into moduleId From Module Where name=modulename;
    End If;

    Insert Into Result(idHost, idModule, datestamp, result) Values(hostId, moduleId, CURDATE(), result);
End;

$$

Delimiter ;

```